VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Měření teplotního profilu vrtu

Borehole temperature profile measurement

2013 Lukáš Šilbach

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Lukáš Šilbach**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2601R004 Měřicí a řídicí technika

Téma: Měření teplotního profilu vrtu
Borehole Temperature Profile Measurement

Zásady pro vypracování:

1. Rozbor současného stavu měření teplot v geotermálních výměnících pro tepelná čerpadla.
2. Analýza způsobů měření teplotního profilu vrtu.
3. Rozbor způsobů sběru dat z čidel komunikujících po sběrnici 1-wire.
4. Návrh a realizace vizualizační aplikace pro vyčítání dat ze sítě čidel.
5. Testování funkčnosti systému a realizace ověřovacího měření.
6. Zhodnocení výsledků.
7. Rozšířený abstrakt v anglickém jazyce v rozsahu 3 strany sumarizující řešení. Rozšířený abstrakt bude přiložen jako jedna z příloh.

Seznam doporučené odborné literatury:

[1] *Studie metodiky měření vlastností horninového masivu v návaznosti na efektivní provoz tepelných čerpadel.* Závěrečná zpráva k projektu MPO Efekt 122142 0234, 2011.
[2] Firemní technická dokumentace pro použitá čidla.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Jiří Koziorek, Ph.D.**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2013

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 1.5.2013

Podpis studenta.....

Lukáš Šilbach

Poděkování

Rád bych zde poděkoval vedoucímu bakalářské práce doc. Ing. Jiřímu Koziorkovi, Ph.D. a také doktorandům za jejich rady a čas, který mi věnovali při řešení dané problematiky.

Abstrakt

Tato bakalářská práce se zabývá návrhem vizualizace pomocí softwarového nástroje Microsoft Visual Studio. Hlavním úkolem této práce je vytvoření vizualizace pro zobrazování a zpracování naměřených dat. Data jsou získávána z digitálních teploměrů umístěných ve vrtech. Naměřená data jsou zobrazována v textových polích a následně je možné uložit je do tabulkového procesoru Microsoft Excel. Takto uložena data můžeme použít pro další zpracovaní.

Klíčová slova

1-Wire, RS-232, ADA-101W, Visual Studio, C#

Abstract

This bachelor thesis deals with visualization using the software tool Microsoft Visual Studio. The main task of this work is to create visualizations for displaying and processing of measured data. Data are obtained from digital thermometers placed in the wells. The measured data are displayed in text boxes and then can be saved in Microsoft Excel spreadsheet. This stored data can be used for further processing.

Key words

1-Wire, RS-232, ADA-101W, Visual Studio, C#

Seznam použitých zkratek

C#	objektově orientovaný programovací jazyk
C, C++	programovací jazyky
CRC	(Cyclic Redundant Control) - Cyklická redundantní kontrola
LSB	(*least significant bit*) - nejméně významný bit
MSB	(most significant bit) - nejvýznamnější bit
RxD	(Received Data) - příjem dat
TxD	(Transmitted Data) - vysílaní dat
GND	(ground) - zem
PWR	(power) - napájení
TTL	(transistor-transistor-logic) - tranzistorově-tranzistorová logika
VDD	napájecí napětí
DQ	(Data Input/Output) - datový vstup a výstup
TH	(temperature high) - horní mez teploty
TL	(temperature low) - dolní mez teploty
DLL	(Dynamic-link library) - dynamicky linkovaná knihovna

Obsah

1. Úvod

Vizualizace se v současně době považuje za velmi rozšířenou v mnohých oblastech. Například v průmyslu, a to především v automobilovém, stavebním, strojírenském, dále pak v medicíně, geografii apod. Můžeme se s ní setkat zejména ve velkých provozech, ale taky v menších, jako jsou např. pekárny, pivovary atd.

Díky vizualizaci můžeme v reálném čase sledovat a řídit technologické procesy. Dále také můžeme monitorovat stavy procesů a kontrolovat jejich kvalitu. Vizualizace nám dává také možnost efektivně reagovat na alarmové stavy a hlášení. Její nedílnou součástí je i možnost uložení a archivace dat. Vizualizace má mnoho výhod, ale jednou z těch předních je ovládání technologického procesu na dálku. Toto ovládání na dálku je možné například pomocí internetu.

Cílem této práce je navrhnout vizualizaci pro kontrolování a sběr informací z teplotních senzorů umístěných v podzemních průzkumných vrtech. Senzory jsou napojené přes sběrnici 1- Wire na převodník. Na tomto převodníku dochází k odečítání hodnot naměřené teploty. Tento současný stav sběru dat je značně nepohodlný, a proto je nutné vytvořit vizualizaci v softwarovém vývojovém nástroji Visual studio. Vytvořená vizualizace bude automaticky v určitém časovém intervalu zobrazovat aktuální teploty, sbírat data ze senzorů a ukládat do tabulek (např. pomocí programu Microsoft Excel). Takto přehledně uložená data bude moci uživatel přenést pomocí přenosných medií, jako jsou flash disky, externí harddisky, paměťové karty a podobná přenosná média pro další zpracování.

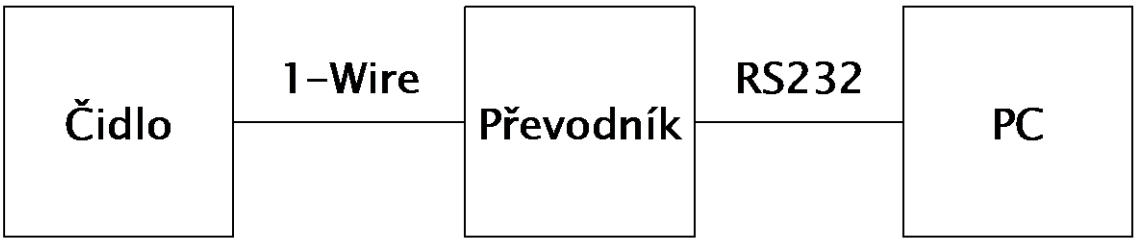
Práce je rozdělena do několika částí. První je teoretická, v níž se budu zabývat postupným popisem jednotlivých částí měřícího řetězce. Ve druhé části se budu zabývat samotným návrhem vizualizačního programu ve vývojovém prostředí Microsoft Visual studio 2008. Zde se pokusím navrhnout vizualizační program pro sběr a archivaci dat, dle zadaných požadavků. V další části budu provádět samotnou realizaci, tzn. vytvořený program připojit na reálný měřící řetězec a uvést jej do provozu. Další důležitou částí je samotné testování programu. Zde budu testovat program, jak v reálném provozu, tak i v krizových situacích. V závěru se pokusím zhodnotit dosažené výsledky a stručně shrnout podstatu bakalářské práce.

2. Teoretický rozbor

2.1. Popis současného stavu

Na obrázku 2. je zobrazené schéma současného měřícího systému pro měření teploty ve vrtu. Základ měřicího systému tvoří čidla DS18B20 od firmy Dallas Semiconductor, která jsou umístěná v hlubinných vrtech a komunikuji přes sběrnici 1–Wire. Tato sběrnice má jeden řídicí obvod (master) a jeden či více ovládaných zařízení (slave). Čidla jsou připojena přes sběrnici 1-Wire na převodník. Jedná se o průmyslový převodník s převodem rozhraní RS232 na 1-Wire rozhraní a naopak. Počítač se k převodníku připojuje přes sériovou linku RS232. V počítači je nainstalován základní testovací program pro vyhledávání zařízeni 1-Wire a zobrazení aktuální teploty. Testovací program byl vytvořen ve vývojovém prostředí Microsoft Visual Studio pomocí jazyka C#.

Obrázek 1. Fotografie kabelu



Obrázek 2. Blokové schéma měřícího systému

Současný program umí v textovém poli zobrazovat nalezené zařízení s adresou zařízení a aktuální teplotou. Jedná se o aplikaci pouze informativního charakteru, nemá v sobě zahrnuty funkce pro ukládání a archivaci dat.

V nově vytvořeném programu, ve vývojovém softwarovém nástroji Visual Studio od společnosti Microsoft, bude dále možné zadávání intervalů měření teploty, zobrazování grafů (tzn. zobrazování hodnot v podobě stále se měnícího grafu v okně Grafy). Dále bude možné naměřené hodnoty teplot ukládat do tabulek v tabulkovém procesoru Microsoft Excel. Tyto tabulky bude možné předat k dalšímu zpracování, a to tak, že je přeneseme na libovolné paměťové médium, nebo je uložíme přímo na pevný disk osobního počítače. Veškeré informace o činnosti, která probíhá v programu, by se měly zobrazovat v informačním okně.

2.2. Microsoft Visual Studio

Microsoft Visual Studio je vývojové prostředí (IDE) od Microsoftu. Může být použito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním spolu s aplikacemi Windows Forms, webovými stránkami, webovými aplikacemi a webovými službami, jak ve strojovém kódu, tak v řízeném kódu na platformách Microsoft Windows, Windows Mobile, Windows CE, .NET, .NET Compact Framework a Microsoft Silverlight.[8]

2.3. C#

C# je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework.

Výhody

- C# je jednoduchý, moderní, mnohoúčelový a objektově orientovaný programovací jazyk.
- Jazyk a jeho implementace poskytuje podporu pro principy softwarového inženýrství, jako jsou: hlídání hranic polí, detekce použití neinicializovaných proměnných a automatický garbage collector. Důležité jsou také jeho vlastnosti jako: robustnost, trvanlivost a programátorská produktivita.
- Jazyk je vhodný pro vývoj softwarových komponent distribuovaných v různých prostředích.
- Přenositelnost zdrojového kódu je velmi důležitá, obzvláště pro ty programátory, kteří jsou obeznámeni s C a C++.
- Mezinárodní podpora je též velmi důležitá.
- C# je navržen pro psaní aplikací jak pro zařízení se sofistikovanými operačními systémy, tak pro zařízení s omezenými možnostmi.

- Přestože by programy psané v C# neměly plýtvat s přiděleným procesorovým časem a pamětí, nemohou se měřit s aplikacemi psanými v C nebo jazyce symbolických adres.

Vlastnosti jazyka

Následující popis je založen na specifikaci jazyka C# a dalších dokumentech, které naleznete v sekci Externí odkazy.

- V C# neexistuje vícenásobná dědičnost - to znamená, že každá třída může být potomkem pouze jedné třídy. Toto rozhodnutí bylo přijato, aby se předešlo komplikacím a přílišné složitosti, která je spojena s vícenásobnou dědičností. Třída může implementovat libovolný počet rozhraní.
- Neexistují žádné globální proměnné a metody. Všechny funkce a metody musí být deklarovány uvnitř tříd. Náhradou za ně jsou statické metody a proměnné veřejných tříd.
- V Objektově orientovaném programování se z důvodu dodržení principu zapouzdření často používá vzor, kdy k datovým atributům třídy lze zvenčí přistupovat pouze nepřímo a to pomocí dvou metod get (accessor) a set (mutator). V C# lze místo toho definovat tzv. Property, která zvenčí stále funguje jako datový atribut, ale uvnitř Property si můžeme definovat get a set metody. Výhodou je jednodušší práce s datovým atributem při zachování principu zapouzdření.
- C# je typově bezpečnější než C++. Jediné výchozí implicitní konverze jsou takové, které jsou považovány za bezpečné jako rozšiřování Integerů (např. z 32 bitového na 64 bitový) nebo konverze z odvozeného typu na typ rodičovský. Neexistuje implicitní konverze z typu Integer na Boolean, ani mezi výčtovým typem enum a typem Integer.
- C# neobsahuje a ani nepotřebuje dopřednou deklaraci - není důležité pořadí deklarace metod. [7]

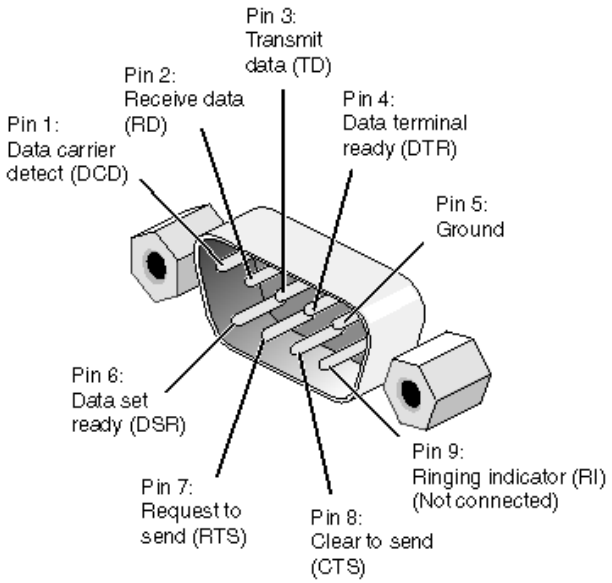
2.4. RS-232

Standard RS-232 se používá jako komunikační rozhraní osobních počítačů a další elektroniky. RS-232 umožňuje propojení a vzájemnou sériovou komunikaci dvou zařízení, tzn., že jednotlivé bity přenášených dat jsou vysílány postupně za sebou (v sérii) po jednom páru vodičů v každém směru.

V současné době se v oblasti osobních počítačů od používání sériového rozhraní RS-232 již téměř definitivně ustoupilo a to bylo nahrazeno výkonnějším Univerzálním sériovým rozhraním (USB). Nicméně v průmyslu je tento standard, především jeho modifikace – standardy RS-422 a RS-485, velice rozšířen a pro své specifické rysy tomu tak bude i nadále. Na běžných sériových portech v PC lze dosáhnout rychlost maximálně 115200bit/s. Ostatní rychlosti jsou

odvozeny dělením 115200bit/s. Jde tedy o řadu 115200bit/s, 57600bit/s, 38400bit/s, 19200bit/s, 9600bit/s, 4800bit/s, 2400bit/s.

Standard definuje asynchronní sériovou komunikaci pro přenos dat. Pořadí přenosu datových bitů je od nejméně významného bitu (LSB) po bit nejvýznamnější (MSB). Počet datových bitů je volitelný, obvykle se používá 8 bitů, lze se také setkat se 7 nebo 9 bity. Logický stav „0“/„1“ přenášených dat je reprezentován pomocí dvou možných úrovní napětí, které jsou bipolární a dle zařízení mohou nabývat hodnot ±5 V, ±10 V, ±12 V nebo ±15 V. Nejčastěji se používá varianta při které logické hodnotě 1 odpovídá napětí −12 V a logické hodnotě 0 pak +12 V. Základní tři vodiče rozhraní (příjem RxD, vysílání TxD a společná zem GND) jsou doplněny ještě dalšími vodiči sloužícími k řízení přenosu (vstupy DCD, DSR, CTS, RI, výstupy DTR, RTS). Ty mohou a nemusí být používány (zapojeny), nebo mohou být použity pro napájení elektronických obvodů v zařízení, jako je například počítačová myš. Výstupní elektronika je vybavena ochranou proti zkratu, kdy po překročení proudu 20 mA proud již dále neroste. Na počítači bývá linka RS-232 vyvedena pomocí konektoru D-Sub typu DE-9 M (samec), zařízení se tedy připojuje kabelem s konektorem DE-9 F (samice).[1]



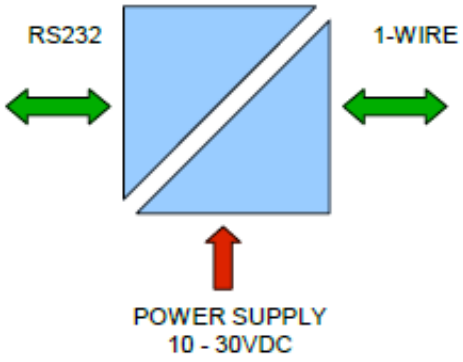
Obrázek 3. Konektor DB-9 s označením pinů [9]

2.5. ADA-101W

Je to průmyslový převodník RS-232 na 1-Wire pro všeobecné použití ADA-101W umožňuje připojení více zařízení s 1-Wire rozhraní, jako jsou systémy pro měření teploty, hodiny reálného času, EPROM, A/D převodníky, atd. na společnou sběrnici 1 - drátu. Pro převod 1-Wire rozhraní na RS-232 rozhraní se používá převodník DS2480B TTL úrovní RS-232. Převodník umožňuje monitorovat anebo ovládat 1-Wire systém přes rozhraní RS-232 v PC vybaveným příslušným softwarem. ADA-101W je vybaven konektorem DB-9 pro připojení k rozhraní RS-232 a svorkovnicí pro napojení 1-Wire sběrnice a napájení. DB-9 konektor pro RS-232 je vyroben jako DCE, která umožňuje připojení převodníku s jiným zařízením vybaveny RS-232. Pro provoz převodníku ADA-101W využívá signály Rx, Tx, GND. Na přední části převodníku se nachází tři LED diody (červená - RX, žlutá - TX a zelená - PWR).

Přenosové rychlosti jsou ve standartním režimu 16,3 kbit/s a maximální rychlost je 142 kbit/s. ADA-101W je určen pro napájení z externího zdroje stejnosměrného napětí, jehož hodnota by měla být v rozsahu od 10V až 30V, a byl dodáván minimální výkon 2W. Převodník také má vestavěnou ochranu proti převrácení polarity zdroj energie. [2]

Obrázek 4. Převodník ADA-101W [2]



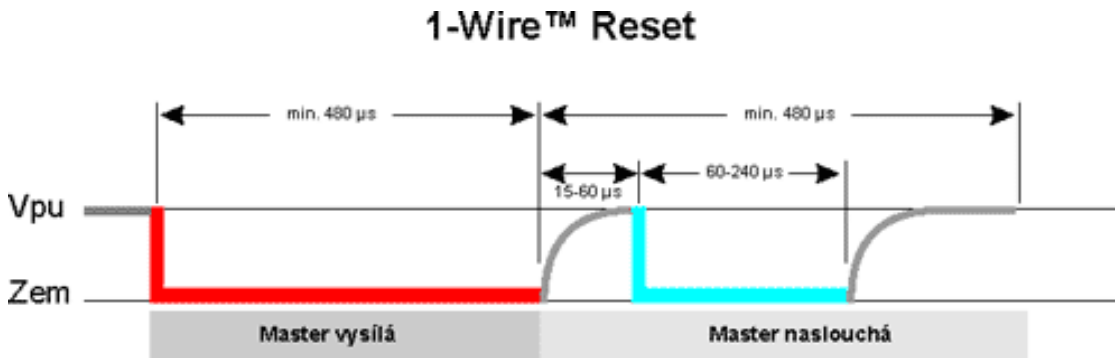
Obrázek 5. Galvanické oddělení [2]

2.6. 1-Wire

Sběrnice 1- Wire navržená firmou Dallas Semiconductor, umožňuje připojit několik zařízení k řídící jednotce prostřednictvím pouhých dvou vodičů. Sběrnice má jeden řídící obvod (master) a jeden či více ovládaných zařízení (slave). Všechny obvody jsou zapojeny jednak na společnou zem, jednak paralelně na společný datový vodič. Tento datový vodič je připojen přes odpor cca 5k na napájecí napětí a "zdvihá" tak sběrnici do log. 1. Přenos dat v 1-Wire síti je asynchronní a poloduplexní. Veškeré informace obíhající v síti jsou účastníky přijímány jako příkazy nebo jako data. Příkazy jsou generovány řídícím členem (master) sítě, provádí různé varianty hledání a adresace známých zařízení, určují aktivitu na síti, řídí přenos dat v síti apod. Standardní rychlost práce 1-Wire sítě 16kbit/s byla vybrána jednak pro zajištění maximální spolehlivosti přenosu dat na velké vzdálenosti, jednak pro přizpůsobení k rychlosti nejrozšířenějších typů mikropočítačů, které musí být v zásadě použity pro řídící členy 1-Wire sběrnice. V síti 1-Wire se používají standardní TTL úrovně signálů a napájení většiny komponent může být z vnějšího zdroje s pracovním napětím od 2,8 do 6 V. Alternativou vnějšího zdroje je parazitní napájení, založené na využití elektrické energie impulzů předávaných po lince dat a akumulované vestavěnou kapacitou. Mimo to mohou jednotlivé komponenty 1-Wire sítí využívat režim napájení po sběrnici dat. [5]

2.6.1. 1-Wire reset

Komunikaci zahajuje vždy master **reset pulsem**. Nejprve "stáhne" datový vodič do log. 0 (uzemní ho) a drží ho na této úrovni minimálně 480 mikrosekund. Pak sběrnici uvolní a naslouchá. Odpor zatím vrátí sběrnici zpět do log. 1. Pokud je na sběrnici připojené nějaké 1-Wire zařízení, tak detekuje tuto vzestupnou hranu a po prodlevě (15 - 60 µs) stáhne sběrnici na 60 - 240 µs k log. 0.

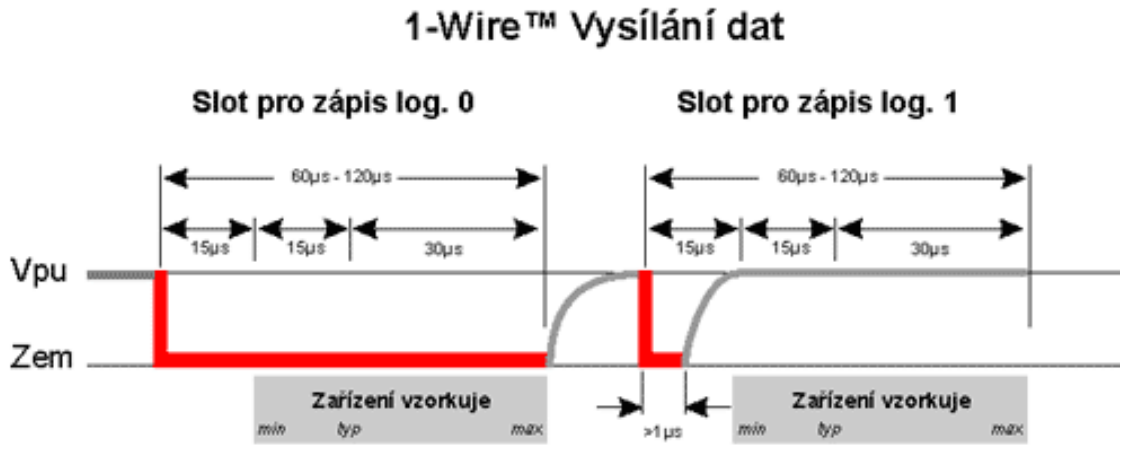


Obrázek 6. 1-Wire reset [5]

2.6.2. Vysílání a příjem dat

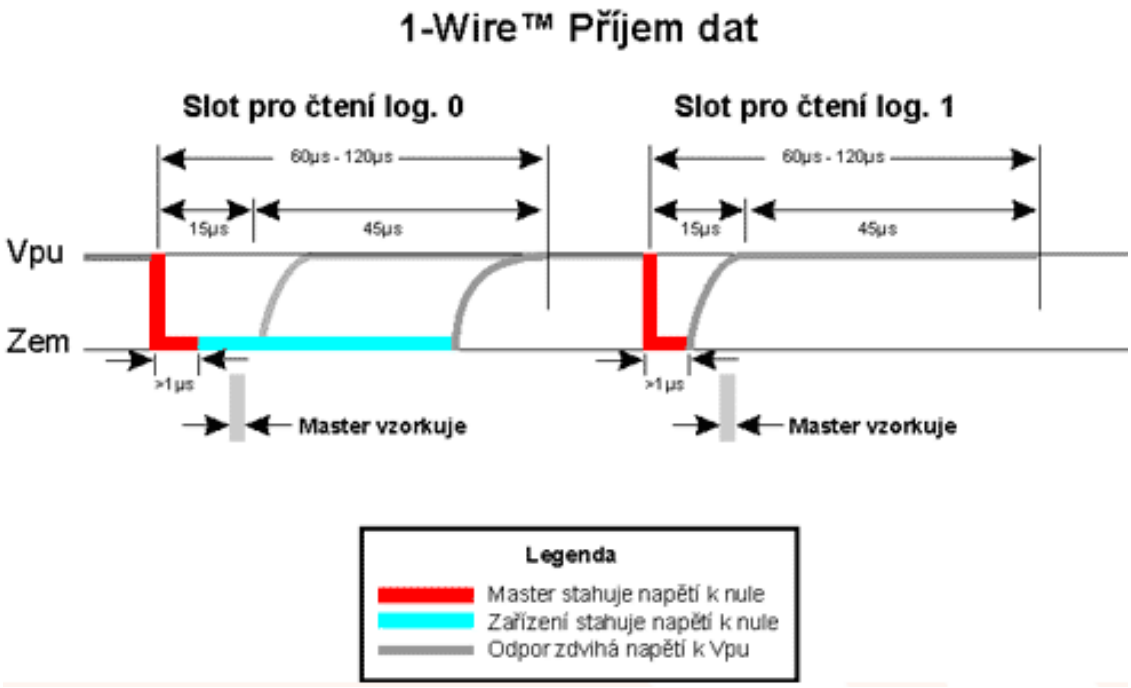
Pokud se zařízení správně ohlásí, může master začít vysílat a přijímat data. Data jsou vysílána v tzv. "time slotech", česky bychom řekli nejspíš v "časových úsecích" nebo v "okénkách". Slot je dlouhý 60 až 120 µs a během jednoho slotu je vyslán, nebo přijat jeden bit informace. Mezi jednotlivými sloty musí být minimálně 1 µs mezera, kdy je sběrnice v klidu.

Existují 4 druhy slotů: Zápis 1, Zápis 0, Čtení 1 a Čtení 0. Zápisové sloty slouží k tomu, aby master vyslal data do zařízení. **Zápis 1** probíhá tak, že master stáhne sběrnici k nule minimálně na 1 µs a nejpozději do 15 µs od začátku ji opět uvolní a ponechá uvolněnou. Zdvihací odpor ji tedy vytáhne k log. 1. **Zápis 0** je o něco jednodušší: Master stáhne sběrnici k 0 a ponechá ji tak po celý slot, tedy min. 60 µs. Zařízení vzorkuje stav na datovém vodiči zhruba 30 µs po začátku timeslotu. [5]



Obrázek 7. 1-Wire vysílání dat [5]

Čtecí sloty opět inicializuje master tím, že stáhne sběrnici k nule na minimálně 1 µs a opět ji uvolní. Po tomto zahájení může zařízení vyslat 1 bit buď tím, že ponechá sběrnici v klidu (log. 1) nebo že ji stáhne (log. 0).

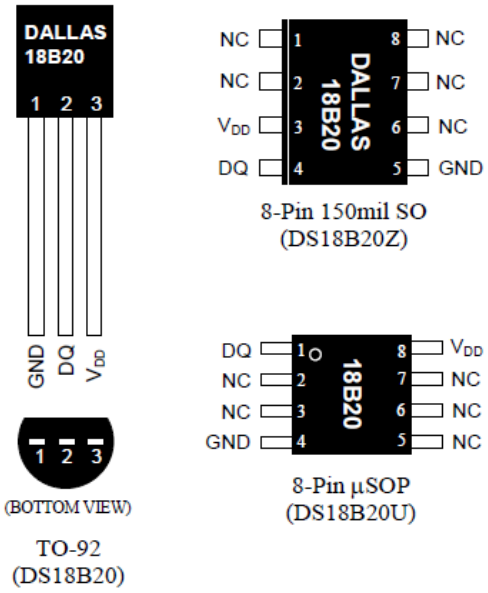


Obrázek 8. 1-Wire příjem dat [5]

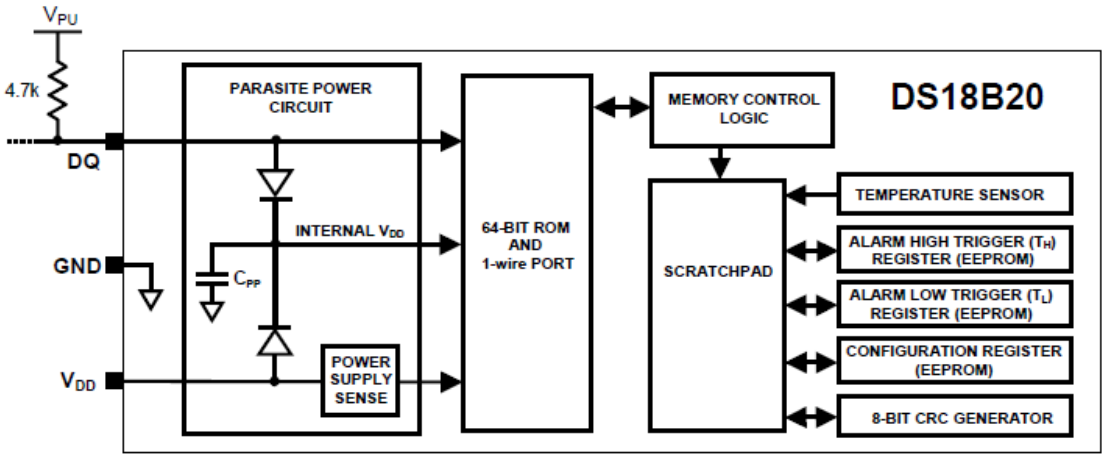
2.7. Čidla DS18B20

Je to čidlo teploty s komunikačním rozhraním 1-Wire. Výstupní data (teplota) jsou v °C a jejich přesnost je ±0,5 °C v rozsahu -10 °C až 85°C a ±2 °C v maximálním rozsahu od -55 °C do 125 °C. Jsou vyráběna v pouzdrech TO92 pro klasickou, tak i v pouzdrech SO, µSOP pro smd montáž. Výsledek převodu teploty je uživatelem nastavitelný na 9, 10, 11 nebo 12 bitů, a tomu odpovídá rozlišení 0,5°C (9bitů), 0,25°C (10bitů), 0,125°C (11bitů), 0,0625°C (12bitů). Výchozí nastavení po zapnutí napájení je 12- bitový převod. Mikroprocesor identifikuje a adresuje zařízení na sběrnici pomocí unikátního 64-bit. kódu. Protože každé zařízení má unikátní kód, může se teoreticky komunikovat s nekonečným počtem zařízení na sběrnici. Čidlo obsahuje tři výstupy, kde dva slouží pro napájení (GND a VDD) a třetí (DQ) slouží pro komunikaci po 1-Wire sběrnici. Napájecí napětí se může pohybovat v rozsahu 3,3–5,5V, proudový odběr je maximálně 1,5 mA. Další vlastností čidla DS18B20 je schopnost pracovat bez externího napájecího zdroje. Napájení je provedeno přes 1-Wire pull-up rezistor pin DQ tehdy, když je sběrnice v log.1. Log.1 sběrnice také nabíjí interní kondenzátor CPP, který nahrazuje napájení při sběrnici v log.0. Tato metoda odvození napájení ze sběrnice se nazývá

„parazitní napájení“. Jako druhá možnost může být čidlo napájeno externím zdrojem přes pin VDD. DS18B20 také umožňuje nastavení spodního a horního teplotního limitu. [3]

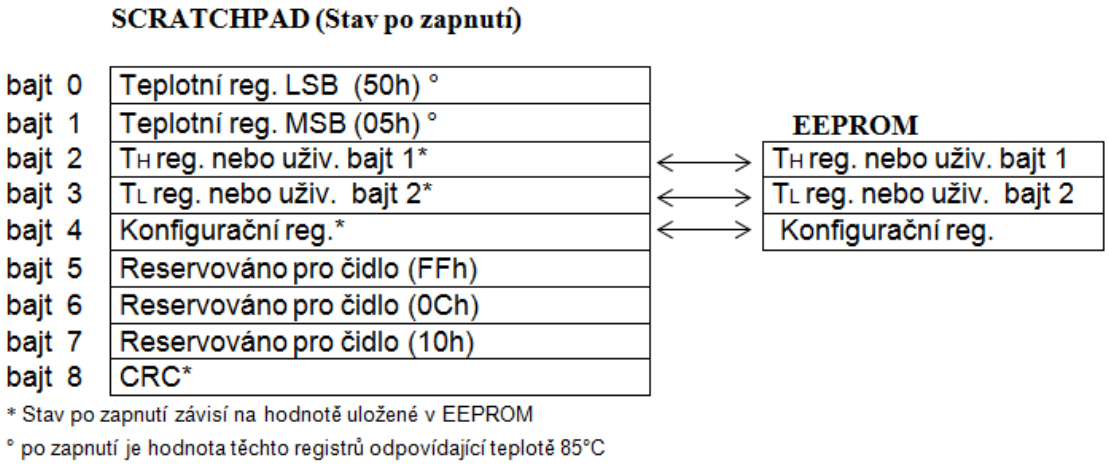


Obrázek 9. Čidlo DS18B20 [3]



Obrázek 10. Blokové schéma DS18B20 [3]

Paměť se skládá z devítibytové scratchpad paměti typu SRAM a paměti EEPROM pro ukládání registrů horní a spodní hranice alarmu teploty (TH a TL) a konfigurační registr. Bajt 0 a bajt 1 z scratchpad obsahuje LSB a MSB registry přiřazené pro převod teploty. Tyto bajty jsou pouze čtecí. Bajty 2 a 3 jsou TH a TL registry. Bajty 4 obsahuje konfigurační registr dat. Bajty 5, 6, a 7 jsou rezervovány pro interní použití zařízením a nemohou být přepsány. Tyto bajty budou při čtení vždy log. 1. Bajt 8 z scratchpadu je pouze pro čtení a obsahuje „cyclic redundancy check“ (CRC) - kód pro bajty 0-7 scratchpadu viz Obrázek 11



Obrázek 11. Scratchpad paměť [3]

2.7.1. ROM příkazy

- Search ROM (Vyhledej ROM) [F0h] – pokud už je komunikace inicializována, master nezná přesný počet zařízení na sběrnici, jejich typ a ani jejich 64bitové ROM kódy. Příkaz Search ROM zjistí 64bitové ROM kódy všech zařízení, která jsou připojena ke sběrnici. Když je na sběrnici pouze jedno čidlo, je možné použít jednodušší příkaz Read ROM.

- Read ROM (Číst ROM) [33h] – příkaz je možné použít pouze tam, kde je na sběrnici přítomné pouze jedno zařízení. Umožňuje řídícímu zařízení přečíst ROM kód slave obvodu bez použití příkazu Search ROM. Pokud je připojeno více zařízení na sběrnici, dojde ke kolizi, jelikož každé zařízení bude vysílat ve stejný okamžik.

- Match ROM (Porovnej ROM) [55h] – tento příkaz, následovaný 64bitovou ROM sekvencí, dovolí master zařízení adresovat konkrétní obvod na sběrnici. Jen čidlo, jehož ROM kód se přesně shoduje s 64bitovou ROM sekvencí, bude reagovat na následující příkazy. Všechna ostatní zařízení budou čekat na další inicializační sekvenci.

- Skip ROM (Přeskoč ROM) [CCh] – dovoluje master zařízení adresovat všechna zařízení bez znalosti jejich ROM kódu. Pokud je zapojeno více než jedno zařízení na sběrnici, následující příkazy se budou týkat všech zařízení. Master může například všem zařízením poslat příkaz pro konverzi teploty najednou. U příkazů pro čtení však bude docházet ke kolizím, protože se budou všechna zařízení snažit odpovědět současně.

- Alarm Search (Hledej alarm) [ECh] – příkaz je podobný příkazu Search ROM. Odpoví na něj ovšem jen čidla, která vykazují překročení teplotních limitů uložených v EEPROM. Master tak má okamžitý přístup k čidlům, která hlásí alarm. [3]

2.7.2. Funkční příkazy

Po adresování ROM příkazem používá master pro komunikaci s čidlem funkční příkazy, které slouží ke čtení a zápisu do scratchpad paměti, ke spuštění teplotní konverze a zjištění způsobu napájení.

- Convert T (Převeď teplotu) [44h] – tento příkaz spustí převod teploty. Čidlo DS18B20 vykoná konverzi, výsledek uloží do dvoubytového teplotního registru scratchpadu a poté zůstane nečinné s minimálním odběrem. Pokud je použito parazitní napájení, musí master do 10 μs vrátit datovou linku na vysokou úroveň, protože čidlo potřebuje dostatek proudu (1,5 mA) k vykonání konverze a je potřeba zajistit, aby se nevybil interní kondenzátor CPP. Jestliže je použito externí napájení čidla, řídicí systém může spustit čtecí slot a číst data příchozí po tomto příkazu. Na sběrnici bude log. 0 po dobu, kdy čidlo provádí konverzi teploty a log. 1, jakmile je převod dokončen.

- Write Scratchpad (Zápis scratchpad) [4Eh] – zapíše tříbytovou informaci do scratchpadu. První byte je zapsán do registru TH, druhý do registru TL (teplotní limity pro spuštění alarmu) a třetí do konfiguračního registru.

- Read Scratchpad (Čtení scratchpad) [BEh] – tento příkaz čte postupně obsah scratchpadu.
  Čtení je zahájeno nultým bytem (LSB teplotního registru) a pokračuje až k osmému bytu (CRC). Master zařízení může kdykoliv ukončit čtení, pokud nechce číst celý scratchpad.

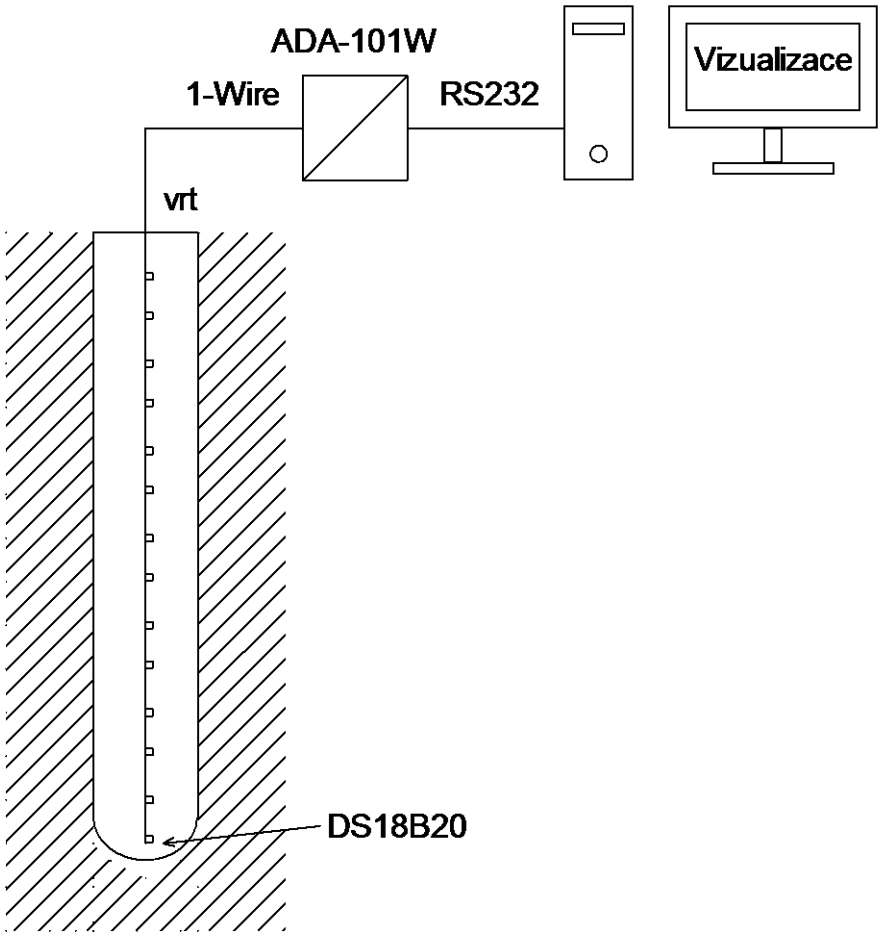
- Copy Scratchpad (Kopírování scratchpad) [48h] – příkaz zkopíruje nastavení teplotních alarmů a konfiguračního registru (byty 2, 3 a 4) do EEPROM paměti. Pokud je použito parazitní napájení, musí master zařízení do 10 μs vrátit datovou linku na vysokou úroveň na nejméně 10 ms, aby proběhlo zapsání do EEPROM správně (podobně jako při konverzi teploty).

- Recall E2 [B8h] – tento příkaz zkopíruje byty TH, TL (teplotní limity pro spuštění alarmu) a konfiguračního byte z EEPROM paměti do scratchpadu. Master může zahájit čtení a na sběrnici bude log. 0 po dobu kopírování a log. 1,

- READ POWER SUPPLY [B4h] – Master vyšle tento příkaz pro určení druhu napájení čidla. To zjistíme tak, že po vyslání příkazu *Read Power Supply* vyšle master „čtecí časový úsek“ – při parazitním napájení bude „čtený časový úsek“ log. 0 a při externím napájení bude log. 1. [3]

3. Návrh řešení

Tato kapitola se zabývá navrhem řěšení, zapojení systému. Na obrázku 12. je zobrazeno situační schéma měřícího systému. V počítači je nainstalován vizualizační software Visual Studio od společnosti Microsoft .



Obrázek 12. Situační schéma měřícího systému

3.1. Knihovna DLL

Dynamická knihovna (DLL) je spustitelný soubor, který pracuje jako sdílená knihovna funkcí. Dynamické propojení umožňuje procesu volat funkci, která není součástí jeho spustitelného kódu. Spustitelný kód pro funkci je umístěn v knihovně DLL, která obsahuje jednu nebo více funkcí, které jsou kompilovány, propojeny a skladovány odděleně od procesů, které je používají. Knihovny DLL také usnadňují sdílení dat a prostředků. Více aplikací může mít současně přístup k obsahu jedné kopie knihovny DLL v paměti.

Výhodou DLL knihovny je využívání méně prostředků tzn. Při použití stejné knihovny funkcí pro více programů. Knihovna DLL může snížit zdrojový kód, který je načten na disku a ve fyzické paměti. To může výrazně ovlivňování výkonu nejen programů spuštěných v popředí, ale také dalších programů spuštěných v operačním systému Windows. DLL knihovny mají modulární architekturu, která pomáhá vyvíjet rozsáhlé programy složené z několika modulů. Tyto moduly lze dynamicky načíst při spuštění programu. V případě aktualizace nebo opravy knihovny DLL není nutné knihovnu znovu připojovat a všechny programy spojené s touto knihovnou budou využívat aktualizovanou knihovnu bez toho, abychom museli zasahovat do programu jednotlivě.

Nevýhodou knihoven v případě smazání (chybové hlášení a aplikace se nespustí). Další nevýhodou je delší přístup k procedurám a funkcím z knihoven. [6]

Přistupování ke knihovnám ve Visual studiu bylo provedeno pomocí přidání referencí do vytvářeného programu. Aby bylo možné volat samotné funkce, které jsou v dll knihovně, musí se na začátek kódu programu přidat slovo using s příslušným odkazem na knihovnu.

Ukázka kódu:

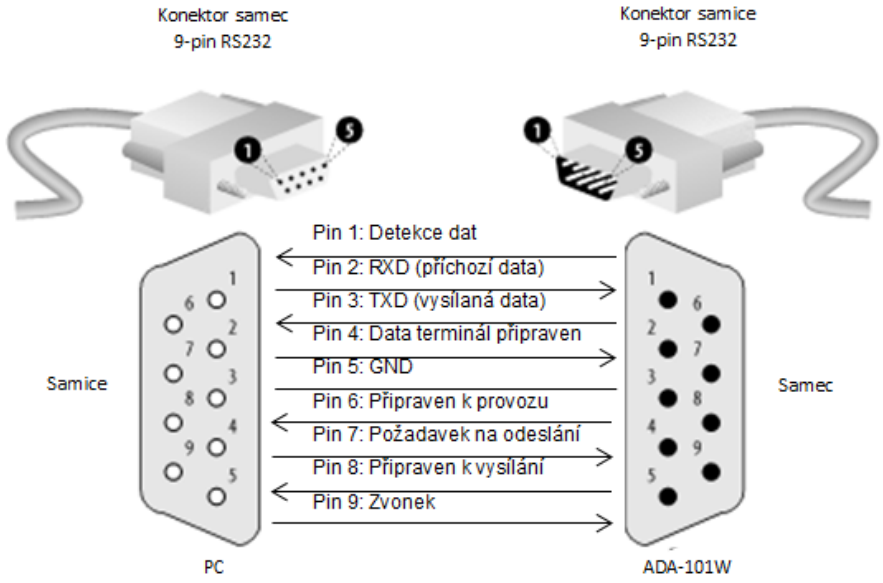
```
using DalSemi.OneWire;
using DalSemi.OneWire.Adapter;
using DalSemi.Utils;
```

3.2. Zapojení RS232

Komunikace mezi převodníkem a počítačem probíhá po sběrnici RS232. Na obrázku 13. je vidět schéma propojení počítače a převodníku s použitými piny Rx, Tx a GND. Propojení je realizováno tři žílovým vodičem. Jako Master je v tomto případě počítač a Slave je převodník ADA-101W. Jednotlivé piny konektorů jsou propojeny následovně: pin Rx Master se propojí s pinem Tx Slave, pin Tx Master se propojí s pinem Slave Rx a piny označené GND obou konektorů se propojí.

Počítač		ADA-101W	
RS232 Konektor		RS232 Konektor	1-WIRE Konektor
Rx		Tx	GND
Tx		Rx	1-W
			VDD
			Vss+
GND		GND	Vss-

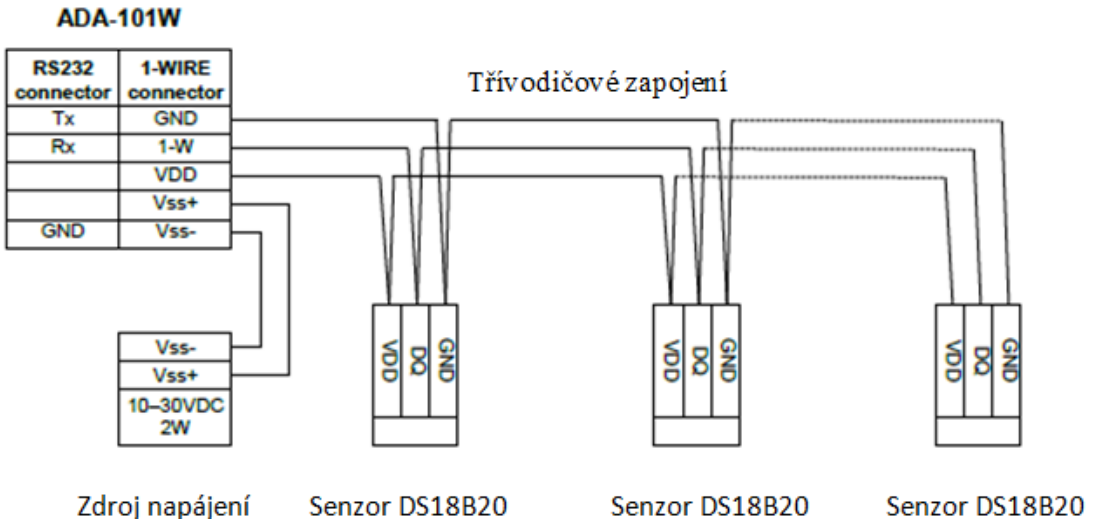
Obrázek 13. Schéma zapojení RS232



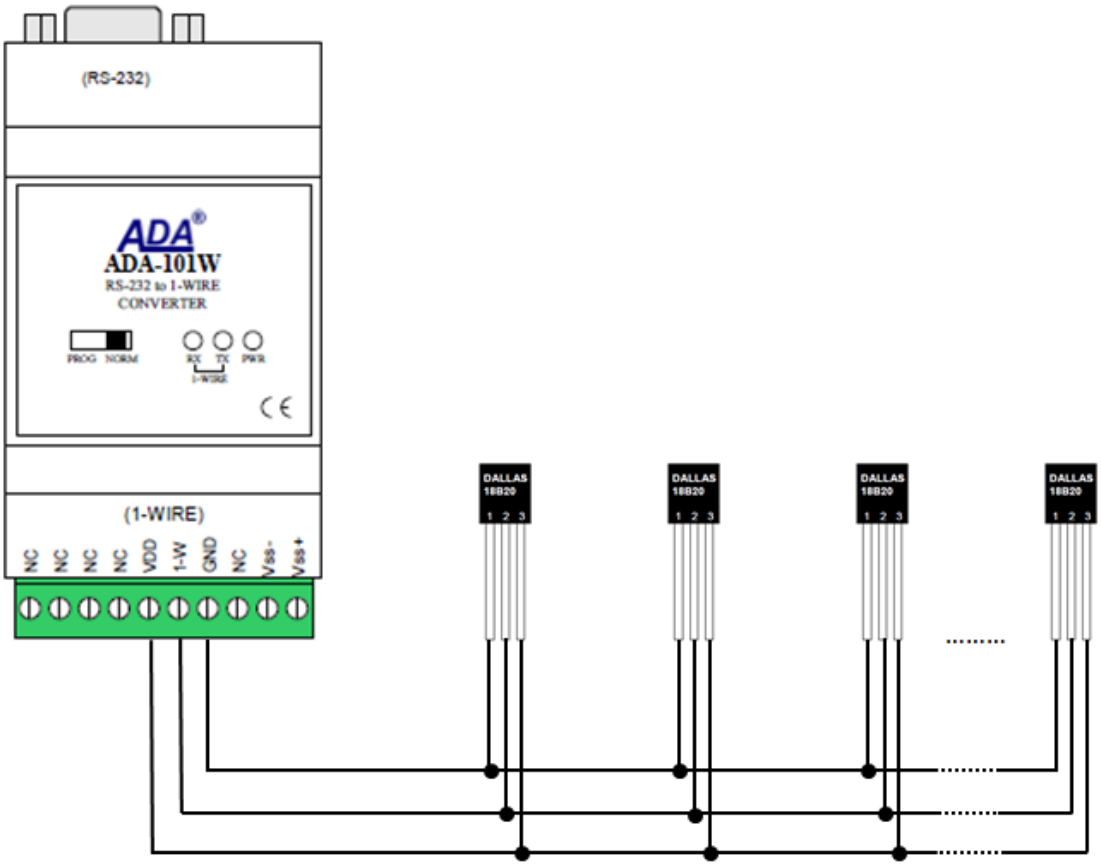
Obrázek 14. Konektory RS232 s popisem pinů [10]

3.3. Zapojení 1- Wire

Komunikace mezi převodníkem a teplotními čidly probíhá po sběrnici 1-Wire. Na obrázku 15 je schéma zapojení třívodičové sítě 1-Wire. V tomto případě je Masterem převodník ADA-101W a jako Slave jsou teplotní čidla DS18B20. ADA-101W je určen pro napájení z externího zdroje stejnosměrného napětí, jehož hodnota by měla být v rozsahu od 10V až 30V, a byl s minimálním výkonem 2W. Slave (DS18B20) se připojí k Master (ADA-101W) pomocí třívodičového vedení. Jeden vodič se připojí na pin 1-W Master a propojí se Slave na pin DQ, druhý vodič je zemnící, jimž se propojí jednotlivé Slave pin GND a připojí se k Master na pin GND a třetí vodič propojí piny označené VDD(napájení). Z obrázku vyplývá, že teplotní čidla jsou k převodníku připojena paralelně a jedná se o tzv. kořenovou topologii.



Obrázek 15. Schéma zapojení 1-Wire [2]



Obrázek 16. Reálné zapojení 1-Wire

4. Realizace

Následující body se budou zabývat samotnou realizací projektu. Pro ten byl vybrán již zmíněný návrh programu pro sběr dat ze senzorů. Nejprve byl nastíněn celkový postup realizace, které bude v této kapitole věnováno nejvíce pozornosti. Pak je třeba se zaměřit na grafickou stránku programu.

Pro realizaci programu bylo nutné, pečlivě nastudovat dokumenty k převodníkům a senzorům. Ty pocházejí z internetových portálů výrobců a univerzitní knihovny. Po prostudování veškeré literatury bylo zahájeno vytváření základních struktur programu samotného.

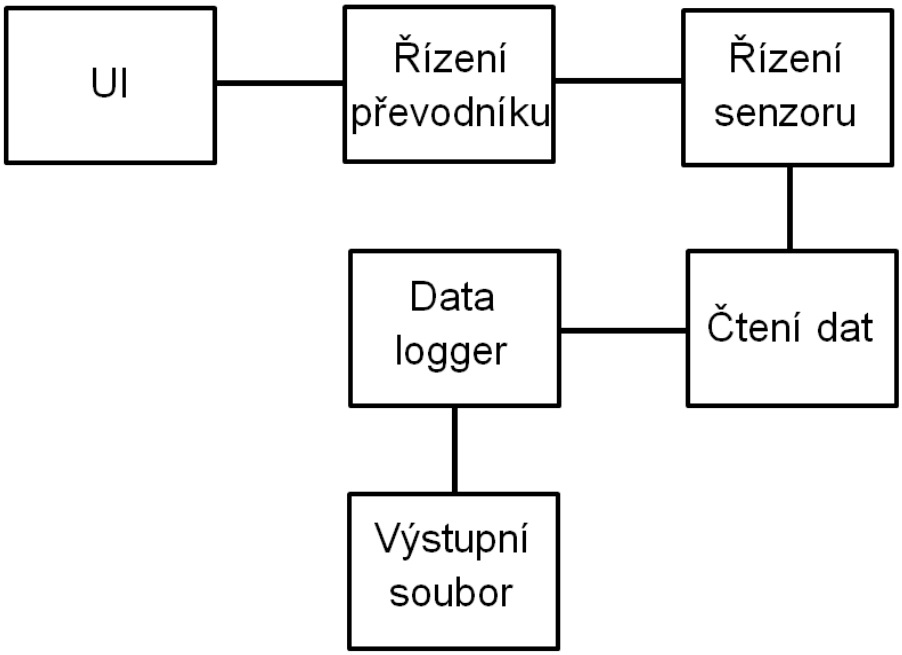
V prvotní fázi bylo vycházeno z parametrů použitých zařízení převodníku DS2480B a senzorů DS18B20. Byla použita knihovna OneWireLinkLayer, která se dá stáhnout na internetových stránkách výrobce Dallas Semiconductor-Maxim. Zde se nacházejí metody realizující sériový port DalSemi OneWire Adapter a metody pro výpočty DalSemi Utils. Tyto metody byly aplikovány do mého programu pro sběr dat ze senzorů. Aby bylo zabráněno složitějšímu nastavování parametrů sériového portu v jiných standardních knihovnách, byla použita výše zmíněná knihovna OneWireLinkLayer. Pro ukládání naměřených dat ze senzorů bylo využito tabulkového procesoru Microsoft Excel. Aby byla zjednodušena komunikace mezi mým nově vytvořeným programem a tabulkovým procesorem Excel, byla použita systémová knihovna Microsoft Excel Library. Ta je stálou součástí vizualizačního vývojového prostředí Visual Studio od společnosti Microsoft.

V pozdější fázi realizace bylo nutné provést jednotlivé seznámení s danými komponenty pro samotnou realizaci programu. Byla tedy vytvořena komponenta tabulkového procesoru Microsoft Exel, pro archivaci naměřených hodnot teploty. Dále bylo nutné vyzkoušet funkčnost této komponenty a seznámit se s jejími chybami a nedostatky. Bylo také potřebné se důkladně seznámit s komponentou grafu a tabulky.

Realizace programu byla provedena podle základního blokového schématu viz. Obrázek 17.

Program se skládá z těchto částí:

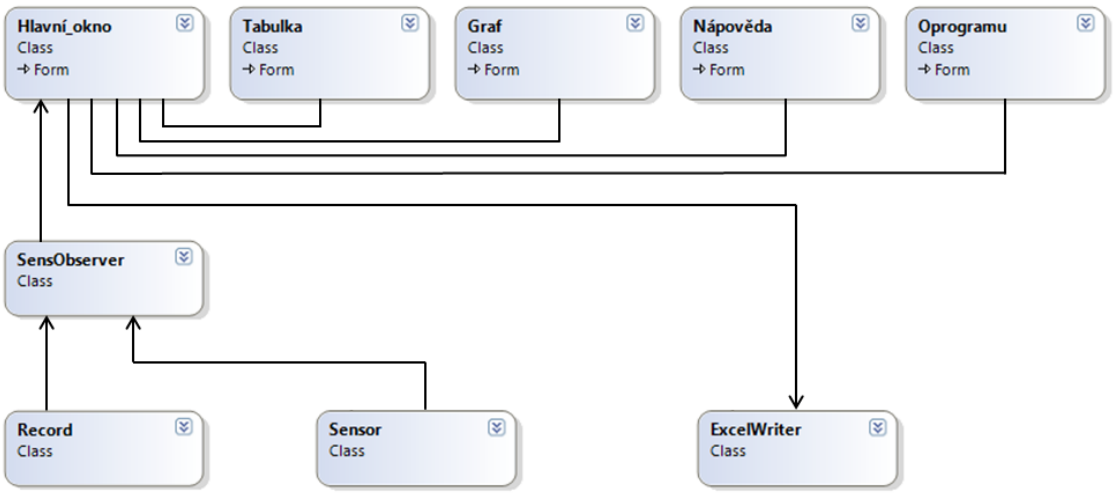
- UI- uživatelské rozhraní
- Řízení převodníku
- Řízení sensoru
- Čtení dat
- Data logger
- Výstupní soubor



Obrázek 17. Blokové schéma programu

4.1. Class diagram

Na obrázku 18. je vidět struktura celého programu vytvořena v class diagramu. Je rozdělena do několika tříd. Ve třídě Hlavní okno jsou obsaženy prvky pro ovládání celého programu, v druhé třídě Graf je prvek pro zobrazování průběhu hodnot teploty v grafu, v další třídě Tabulka je prvek pro zobrazování hodnot v tabulce, v třídě Nápověda je nápověda k programu a jeho ovládání a nakonec v třídě O programu jsou informace o programu. Těchto pět tříd se týká oken programu. Další třídy se budou týkat samotného měření teploty. Třída Record obsahuje vlastnosti, které získávají aktuální čas a hodnotu teploty. Sensor třída má na starost získávání sériového čísla senzoru. SensObserver třída obsahuje metody pro otevření/zavření sériového portu, nastavení adapteru, metody pro nalezení všech připojených senzoru a metodu pro přečtení hodnoty teploty ze senzoru.



Obrázek 18. Class diagram

4.2. Metody programu

Ve třídě ExcelWriter se nacházejí jednotlivé metody. První z nich je metoda WriteHeader, ta slouží pro zapsání hlavičky do Excelu (tzn.: do prvního sloupce prvního řádku vypíše slovo „čas“. Do dalších sloupců v tomtéž řádku vypíše sériová čísla nalezených senzorů). Další metoda slouží pro zapisování dat do Excelu. Tato metoda se nazývá WriteData a její úkolem je zapsat do prvního sloupce čas měření. Do dalších sloupců se pak díky této metodě zapisují veškeré naměřené hodnoty teplot. Poslední metodou v této třídě zůstává SaveFile. Ta slouží k samotnému uložení dat do Excelu podle předem definovaného názvu.

V další třídě Tabulka se nacházejí také jednotlivé metody programu. Jsou velmi podobné jako u třídy ExcelWriter s tím rozdílem, že zde se data nezapisují do tabulkového procesoru Excel, ale do tabulkové komponenty. Jediná metoda nenacházející se v této třídě, je metoda pro ukládání naměřených hodnot. Důvodem je informativní charakter této komponenty.

Graf je další neméně důležitou třídou, o které je potřeba se zmínit. V této třídě se nachází metoda pro generování náhodných barev pro křivky grafu. Další metodou je přidání křivky do kreslící plochy. V této metodě se nastaví titulek, popis os, rozsahy apod. Pro zobrazení bodů na kreslící ploše je zde metoda pro přidání bodů do kreslící plochy.

Ve třídě SensObserver je hned několik jednotlivých metod. Metoda pro připojení a odpojení převodníku. Dalšími metodami, které jsou nejdůležitější v celém programu, se budou zabývat podkapitoly 4.3 Metoda nalezení senzorů a 4.4 Metoda měření teploty.

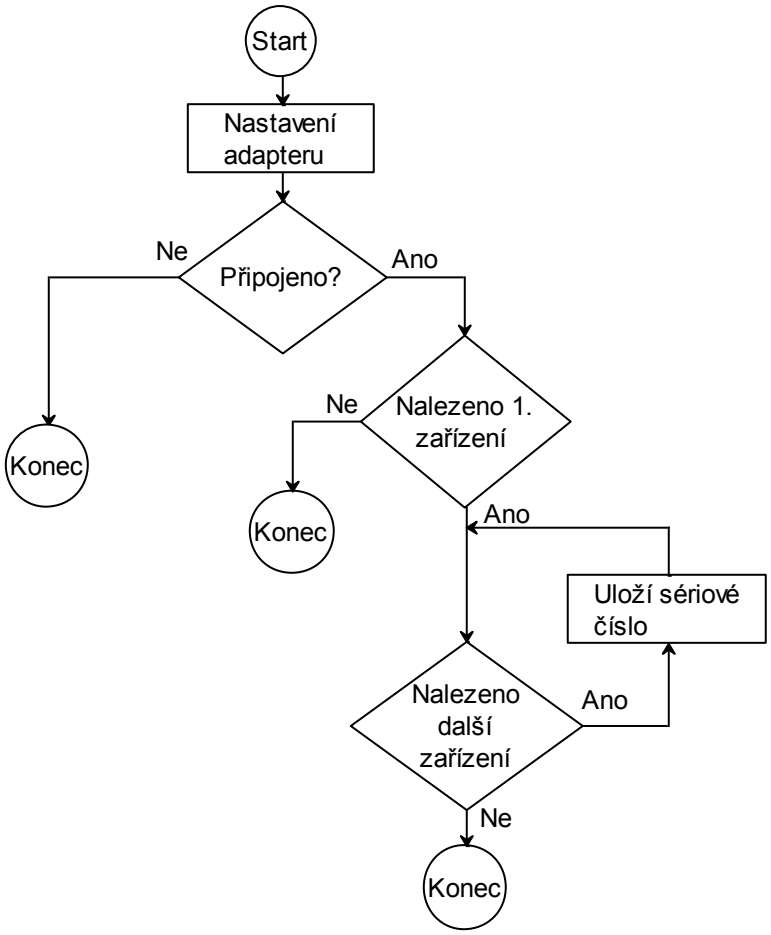
4.3. Metoda nalezení senzorů

Jelikož byl tento program založen na principu analýzy, získávání naměřených dat a jejich archivaci z určitého zdroje informací (zdrojem informace byla zde myšlena hardwarová část pro měření fyzikálních hodnot včetně převodu informace), jenž mohl být fyzicky parametrizován ať rozšířením vlastní struktury zdroje informací, čí změnou celého zdroje informací. Důsledkem toho také docházelo ke změně informací o počtu konkrétně neidentifikovaných připojených zařízení na sběrnici. To sebou neslo i problém s rozpoznáváním původu příchozí informace proudící po sběrnici ke zpracování.

Aby byl eliminován tento problém a mohl být jasně definován původ informací proudících sběrnicí ke zpracování, bylo potřeba identifikovat a označit všechna připojená a aktivní zařízení, která se na sběrnici nacházejí, a která nám data předávají k dalšímu zpracování.

To vedlo k návrhu specifické metody, vytvořené za účelem analyzování komunikační sběrnice, na které bylo očekáváno připojení jednoho či více měřicích zařízení. Následně byla provedena identifikace těchto konkrétních připojených zařízení za pomocí vyčtení sériového čísla z každého komunikujícího zařízení. Poté proběhlo přiřazení této identity do seznamu připojených zařízení. O této skutečnosti byl uživatel následně informován krátkým výpisem informací v informačním okně programu.

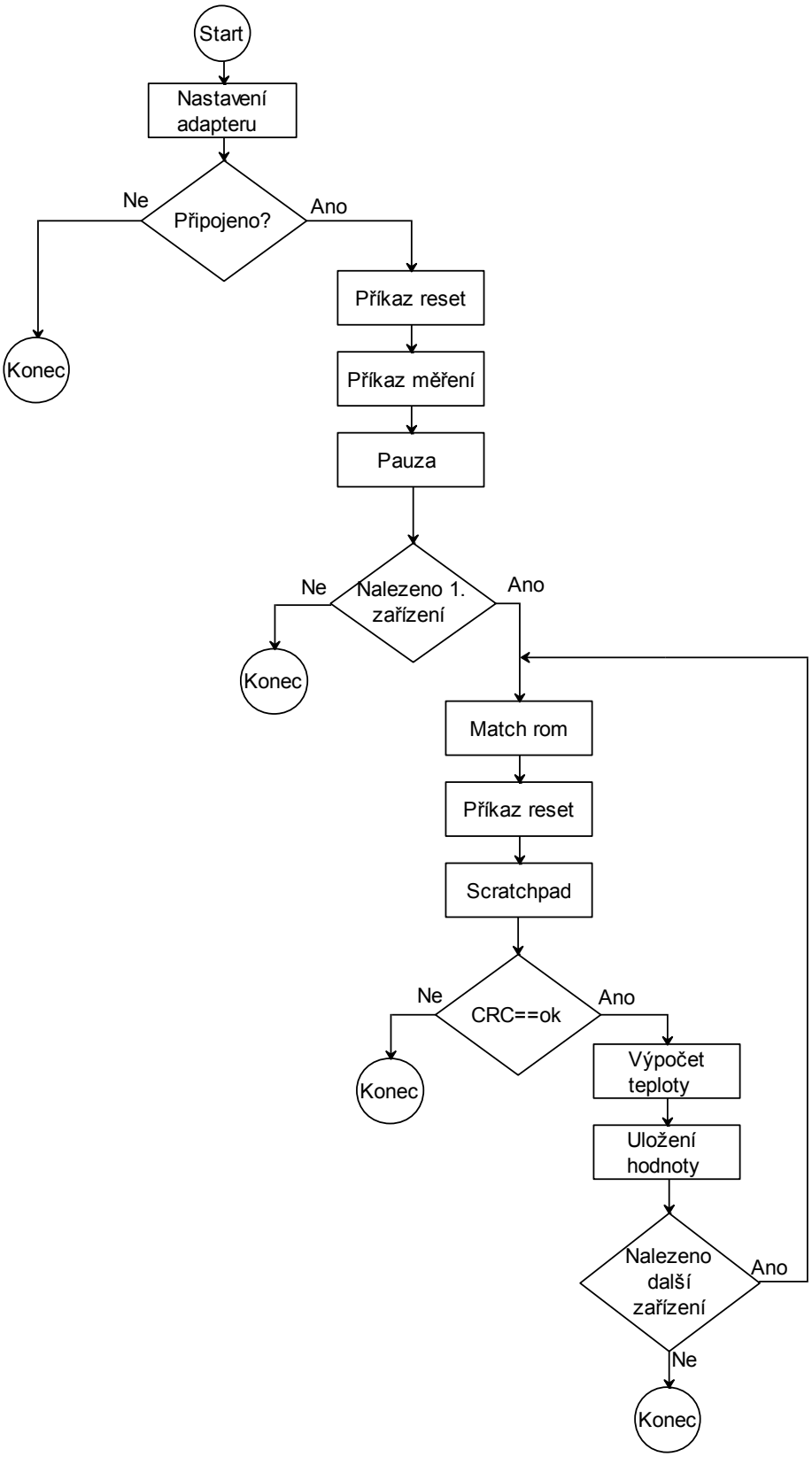
Při volání této metody došlo v první řadě k nastavení adaptéru a konfigurace jeho vnitřních vlastností tak, aby bylo možno navázat komunikační spojení s připojeným zařízením na sběrnici. Korektní nastavení těchto parametrů nám zaručí kladnou odezvu při kontrole připojení ke sběrnici. Následovalo vyhodnocení stavu obsazenosti sběrnice připojeným zařízením, což vedlo k nalezení prvního připojeného zařízení pomocí odezvy zařízení. Pokud byla tato kontrola pozitivní a k dispozici byly informace o nalezeném prvním zařízení, byla provedena opětovná kontrola výskytu následujícího zařízení na sběrnici, která se cyklicky opakovala do nalezení konečného počtu zařízení připojených ke sběrnici.



Obrázek 19. Vývojový diagram metody nalezení senzorů

4.4. Metoda měření teploty

Při volání této metody došlo stejně jako v předchozí metodě nejprve k nastavení adaptéru a konfiguraci. V případě kladné odezvy na kontrolu připojení ke sběrnici následoval sled příkazů důležitých pro správnou funkci měření teploty. Jako první příkaz byl proveden reset, který byl potřebný k vyresetování sběrnice 1-Wire, hned vzápětí následoval příkaz convert pro zahájení měření teploty pro všechny senzory na sběrnici. Následovala krátká pauza pro převod teploty v závislosti na rozlišení v tomto případě 1200ms. Poté došlo k vyhodnocení stavu obsazenosti sběrnice připojeným zařízením, což vedlo k nalezení prvního připojeného zařízení pomocí odezvy zařízení, pokud byla tato kontrola pozitivní a byla možnost komunikovat se senzory na sběrnici. Následným příkazem match rom bylo provedeno navázání komunikace s jedním senzorem, další příkaz provedl reset 1-Wire sběrnice a nakonec došlo k přečtení scratchpad paměti senzoru s příslušnou hodnotou teploty. Korektní příjem dat byl zkontrolován pomocí crc kódu, v důsledku kladné kontroly byl proveden výpočet teploty a v dalším kroku došlo k uložení hodnoty teplot. Poté byla provedena opětovná kontrola výskytu následujícího zařízení na sběrnici, která se cyklicky opakovala do nalezení konečného počtu zařízení připojených ke sběrnici.



Obrázek 20. Vývojový diagram měření teploty

4.4.1. Výpočet teploty

Vypočtení teploty probíhá podle následujících vzorců:

- Kladná teplota

  Nejprve bylo provedeno přetypování horních částí bitů na integer a posunutí o 8 bitů vlevo
  teplota = ((int)(packetD[2])) << 8;

  Výsledek předchozího kroků, byl sečten s přetypovanou dolní částí bitů
  teplota = teplota + ((int)(packetD[1]));

  Výsledná teplota byla vypočtena jako násobení předchozího kroku a koeficientem rozlišení
  vystep = (double)(teplota * 0.0625);

- Záporná teplota

  Převrácení dolních a horních bitů
  invLSB = packetD[1] ^ 0xFF;
  invMSB = packetD[2] ^ 0xFF;

  Převrácené bity horních částí se posunou o 8 bitů vlevo
  teplota = invMSB << 8;

  Výsledek předchozího kroků byl sečten s převrácenou dolní částí bitů
  teplota = teplota + invLSB;

  Výsledná teplota byla vypočtena jako násobení předchozího kroku, koeficientem rozlišení
  a -1
  vystep = (double)(teplota * (0.0625 * (-1)));

4.5. Celkový diagram programu

V této části byla realizována struktura programu pro řízení a parametrizaci hodnot potřebná k získání potřebných dat, která nám tento program vrátí po provedení měření. Tato část byla úzce spjatá s interakcí uživatele, který program ovládá.

Struktura celého programu řízení byla založena na principu postupného definování parametrů získaných uživatelem a zpětnou vazbou systému. Na uživatele byla zde kladena podmínka zadání vstupních parametrů pro výběr komunikačního portu, dále postupným příslušným způsobem spouštění sady ověřovacích podmínek pomocí ovládacích prvků. Přičemž zde probíhá zpětná vazba systému na požadavky zadané uživatelem. Důsledek této akce s korektní zpětnou vazbou vedlo k následné možnosti realizace měření a archivace těchto dat v podobě datových tabulek či grafického znázornění.

Struktura tohoto programu je následující:

- Po spuštění programu systém očekával vstupní parametr výběru komunikačního portu, na kterém by měl dále vyhledávat a komunikovat s připojenými zařízeními

- Zadáním vstupního parametru komunikačního portu a uživatelem vyvolaný požadavek, na připojení ke zvolenému portu stiskem příslušného tlačítka na ovládacím panelu, dojde k pokusu o připojení na zvolený port. Po této akci proběhla kontrola správnosti připojení na požadovaný port a výsledek této kontroly se zobrazil v informačním panelu programu. Po získání kladné odpovědi systému bylo uživateli nabídnuto pokračovat v další parametrizaci konfiguračního řetězce pro měření. Tím se na ovládacím panelu uvolnilo ovládací tlačítko pro nalezení zařízení připojeného ke sběrnici. V případě, že kontrola správnosti připojení ke zvolenému portu dostala negativní výsledek, nebylo možné pokračovat dále v konfiguračním řetězci pro měření a bylo potřeba se od daného portu odpojit a zvolit další port v seznamu a stisknout příslušné tlačítko.

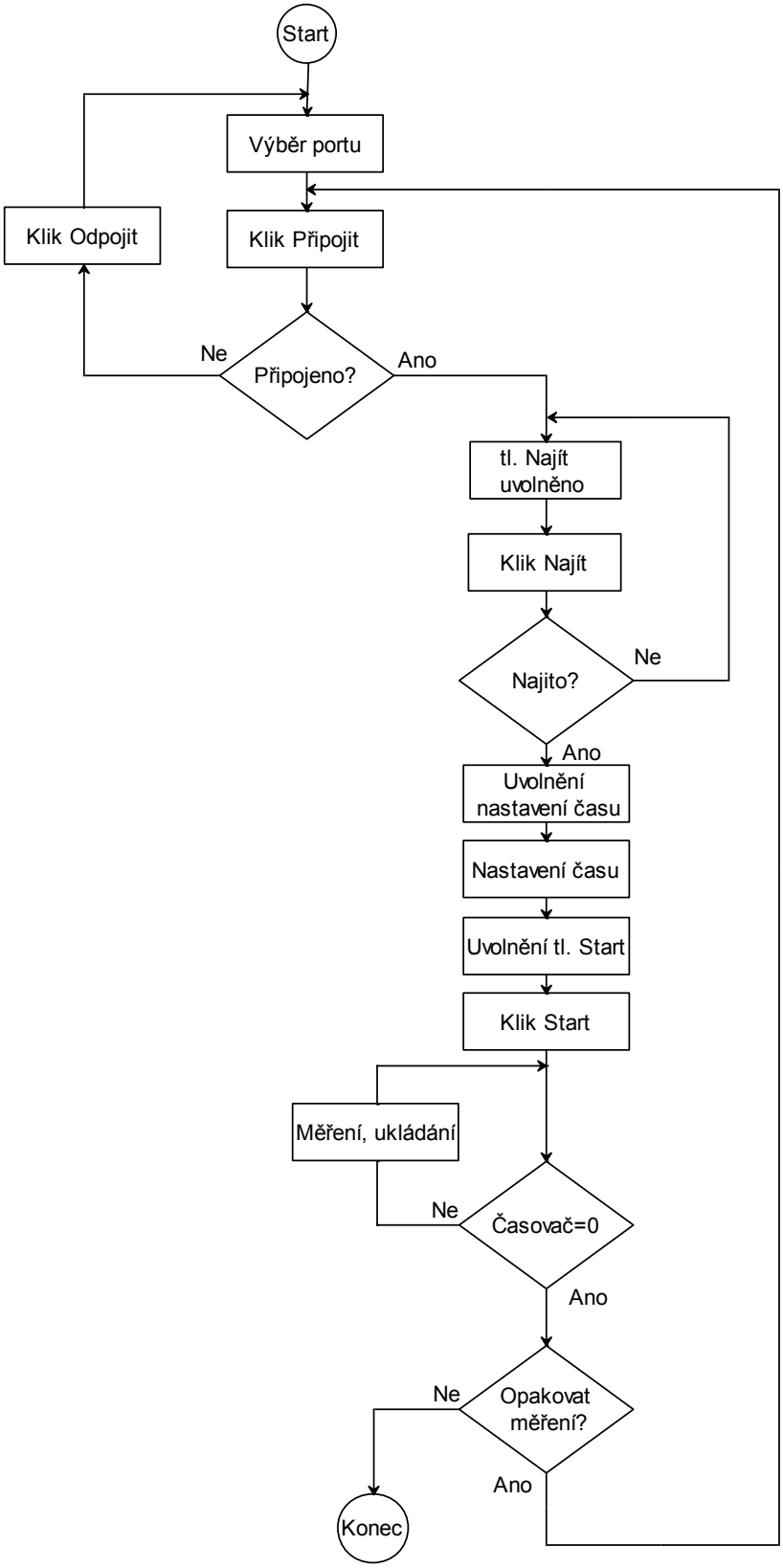
- Kladným výsledkem předchozího kroku, jak bylo zmíněno, bylo uvolnění ovládacího prvku pro zaslání požadavku na vyhledání možného připojeného zařízení na sběrnici. Po zaslání tohoto požadavku uživatelem do systému, byla provedena analýza na detekci možného připojeného zařízení na sběrnici. Zde opět docházelo ke kontrole výsledku analýzy a v případě kladné odpovědi, tzn., že bylo detekováno připojené zařízení, došlo k upozornění uživatele v informačním panelu programu. Tím se dále uvolnila další část konfiguračního řetězce pro nastavení časovače požadované doby měření a získávání dat ze zařízení. V opačném případě, kdy byla odpověď kontroly na detekci zařízení negativní, nebyl povolen další postup v konfiguračním řetězci a bylo nutno se od připojeného portu odpojit.

- Opět v důsledku kladné odpovědi z předchozí kontroly detekce zařízení, bylo nyní umožněno uživateli nastavit dva důležité parametry, kterými ovlivní způsob zpracování

dat a výsledný počet hodnot získaných v průběhu měření. Prvním parametrem byla hodnota celkové délky doby měření a druhým parametrem byla hodnota periody zápisu dat následného měření do tabulek. Tyto parametry byly součástí časovače zobrazeného na ovládacím panelu programu. Uživatel si zde nastavil požadovanou maximální dobu, po kterou bude měření dat probíhat a to ve formátu HH:MM:SS. Druhým parametrem periody zápisu dat nastavil požadovaný krok, s jakým k zápisům hodnot do tabulek bude docházet.

- Nastavením parametrů časovače byla uživateli nabídnuta poslední položka v konfiguračním řetězci, která již sloužila pouze pro zaslání požadavku na spuštění a realizaci měření dat podle výše zmíněného časovače. Během vykonávání tohoto požadavku docházelo k cyklickým zápisům dat do tabulek a to dokud systém nevrátil hodnotu časovače 0, tzn., dokud nedošel časovač do požadované hodnoty pro dobu měření. Poté bylo měření ukončeno a na archivovaná data z měření bylo možně přistoupit rovnou z programu a to ve formě tabulkového souboru, či v grafické podobě grafu.

- Po ukončení měření měl uživatel možnost daný program ukončit nebo mohl měření opakovat.



Obrázek 21. Vývojový diagram programu

4.6. Okna programu

Po kompilaci a zavedení programu do stabilizovaného chodu, bylo důsledkem spuštění programu automaticky vygenerováno uživatelské prostředí zobrazené v samostatném okně programu. Obsah tohoto grafického prostředí programu obsahuje několik funkčně rozdílných částí, které jsou graficky a pozičně uspořádány tak, aby byla práce s programem pro uživatele nejen funkčně účelová, ale také intuitivní.

V těchto oknech bylo použito mnoho komponentů. Mezi nejdůležitější komponenty patří časovače. Jeden z nich byl použit pro odpočet doby měření, druhý byl využit pro přesnou aktualizaci času. Další dva časovače byli využiti pro animaci ukazatelů připojení a komunikaci mezi převodníkem a čidly měřícího řetězce. Důležitými komponenty byla také tlačítka a textová pole. Další komponentou je pole s formátovaným textem. Ta byla použita na výpis sériových čísel a pro výpis právě probíhajících akcí.

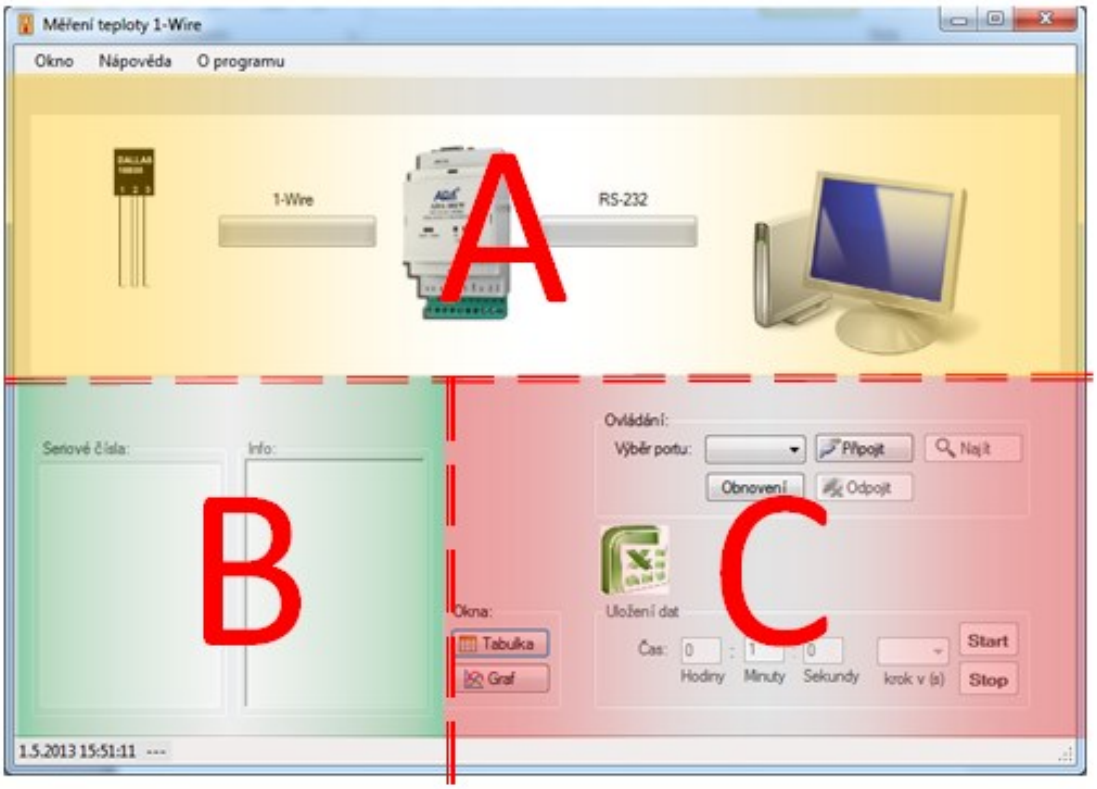
4.6.1. Hlavní okno

Hlavní okno programu bylo uspořádáno do několika sektorů, označených na obrázku *A*, *B* a *C*, které se jednotlivě vyznačují specifickým účelem:

- Pro sektor A je dáno, že jeho primárním účelem bude znázornit stav putování signálu v měřicím řetězci a to z čidla teploty přes sběrnici 1-Wire do převodníku Dallas a dále přes komunikační sběrnici RS-232 do tohoto programu.

- Sektor B je zde za účelem informovanosti obsluhy z hlediska stavu a informací o sériovém čísle připojené jednotky, kde se v levém sloupci vypisují konkrétní sériová čísla pro možnosti kontroly a orientaci o připojené jednotce obsluhy. V pravém sloupci jsou uvedeny veškeré informace o stavu připojení, stavu relace měření, či chybách vyskytujících se během standardního průběhu programu.

- Stěžejním sektorem pro obsluhu je sekce C, sloužící pro veškerou konfiguraci měřeného řetězce a uživatelské ovládání tohoto programu, za účelem vygenerování tabulkového souboru s daty o naměřených teplotách za určitou dobu, nastavenou uživatelem. Uživatel má poté možnost si tato data rovnou prohlédnout jak v tabulkové, tak grafické podobě.



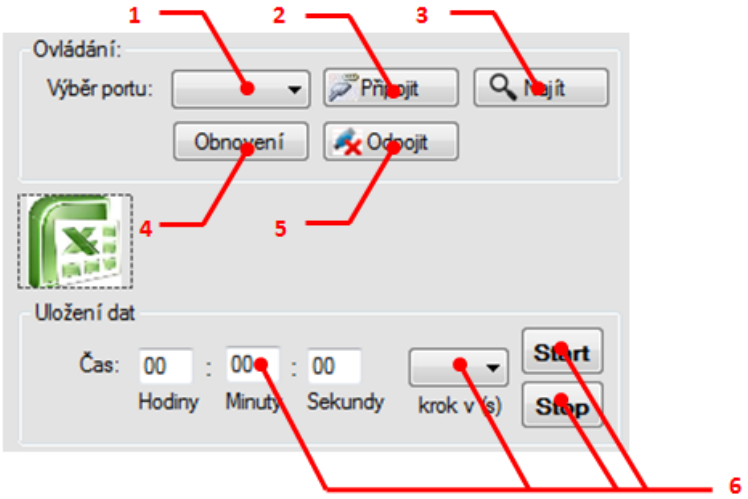
Obrázek 22. Rozčlenění programu do funkčních sektorů

- Po výběru požadovaného portu (1), tento port byl připojen pomocí tlačítka „*Připojit“* (2) v případě potřeby bylo možno seznam nalezených senzorů obnovit tlačítkem „Obnovení“ (4)

- Nyní bylo potřebné zkontrolovat a najít připojené zařízení na požadovaném portu a to tlačítkem *„Najít“* (3)

- Pokud bylo zařízení korektně připojeno, dostane uživatel možnost nastavit časovou konstantu pro pevnou dobu vykonávání měření a získávání dat z připojeného zařízení na zvoleném portu. Uživatel měl možnost nastavit velikost kroku (periody) pro odpočet, respektive zápis získaných dat. Měření bylo možno spustit tlačítkem „*Start*“. Po uplynutí této doby měření tzn. po vynulování časovače, nebo vypnutí měření pomocí tlačítka „*Stop*“ bylo měření a zápis dat ukončeno. (6)

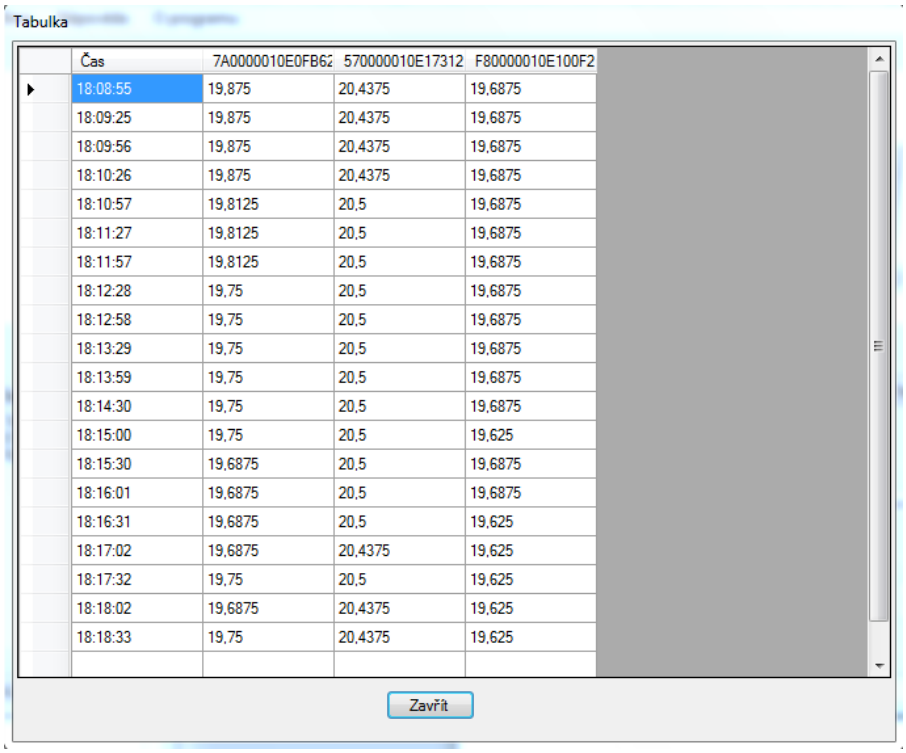
- V případě potřeby ukončení procesu během vykonávání, či v případě resetování připojení má zde k dispozici uživatel tlačítko Odpojit (5)



Obrázek 23. Konfigurace a ovládání aplikace

4.6.2. Okno s tabulkou

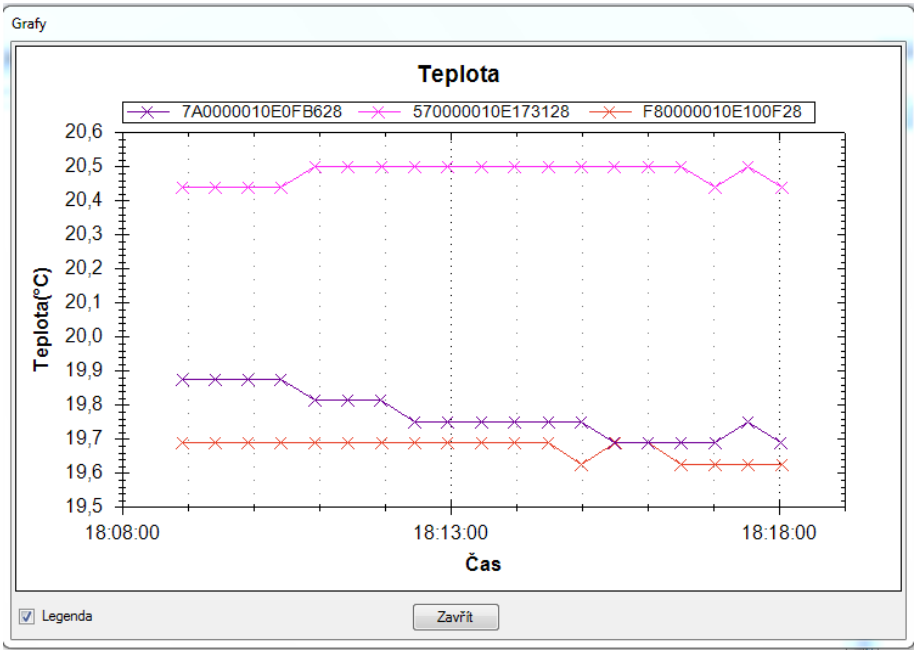
V tomto okně se nachází komponenta pro zobrazování hodnot teploty, které se zároveň ukládají do tabulky Excel.



Obrázek 24. Okno tabulka

4.6.3. Okno s grafem

Toto okno bylo vytvořeno za účelem přehlednějšího zobrazení hodnot teplot. V tomto okně se nachází komponenta pro zobrazování hodnot teplot v grafu.



Obrázek 25. Okno grafy

5. Testy

Pro testování bylo nutné vytvořit kompletní aplikaci. Po jejím vytvoření ve Visual studiu 2008 bylo nutné celou aplikaci otestovat. Toto testování probíhá ve dvou fázích a to ve fázi statické a dynamické. V první fázi (tj. statická) není nutné mít spuštěn program. Statické testování bylo tedy spuštěno ještě před vytvořením prvotní verze programu. Zde byl odhadnut přibližný časový úsek pro vývoj programu. V dynamické fázi testování bylo užito samotného programu. Ten se pak otestoval na různých vstupech a výstupech.

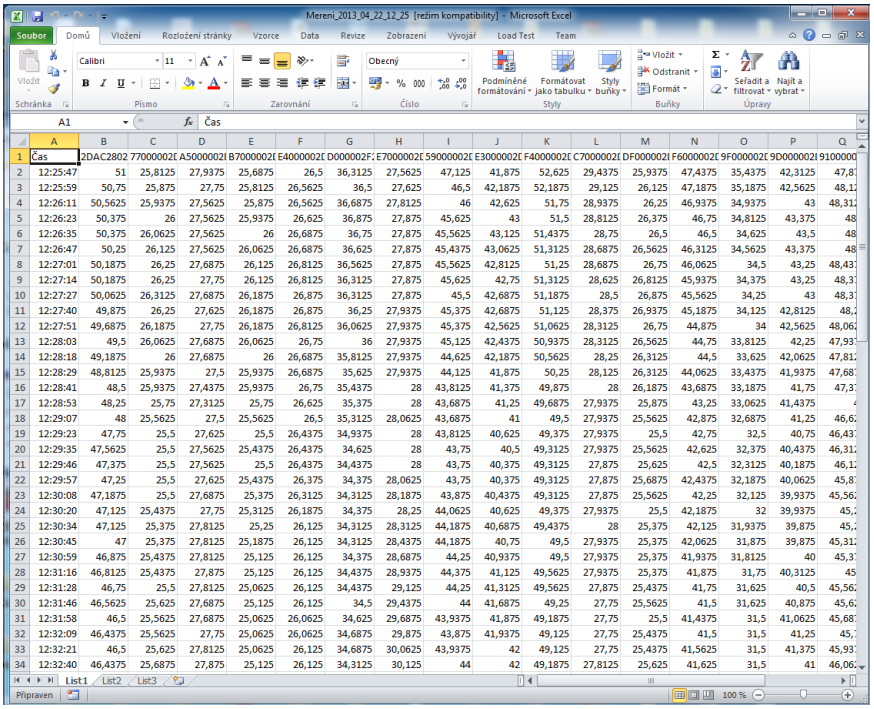
Nejprve bylo nutné vyzkoušet, zda je aplikace schopná se připojit k danému přechodníku na zvolený sériový port. Následně bylo nutné vyzkoušet i její pozdější odpojení. Tato skutečnost slouží k otestování toho, zda po připojení k převodníku dojde k zobrazení informace o připojení na příslušný sériový port ve sloupci označeném jako Info. Stejná skutečnost slouží i k testování toho, zda se po odpojení převodníku zobrazí ve sloupci Info informace o jeho odpojení.

K otestování základních funkcí aplikace byl nejprve použit měřící řetězec. Ten byl zapojen v následujícím pořadí: počítač, následoval převodník a poté samotné senzory v počtu 3 kusů na délce kabelu zhruba 5 metrů. V tomto počtu senzorů a při této délce kabeláže se ukázalo, že aplikace fungovala bez problému. Došlo k výpisu sériových čísel v levém sloupci označený jako sériová čísla. Měření teploty bylo přesné a data se ukládala do předem stanovených tabulek Excel. Zde byla kontrolována správnost uložení sériových čísel. Následně byla na těchto číslech kontrolována správnost přidělení hodnot naměřené teploty.

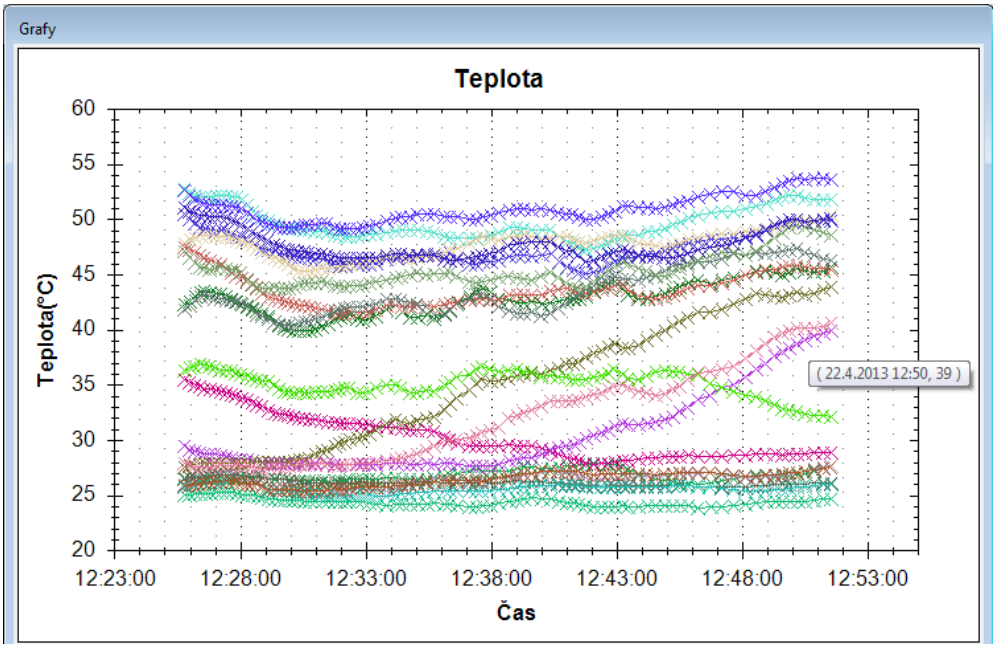
Poté došlo na testování krizových situací. Krizovými situacemi je myšleno odpojení napájecího kabelu nebo přerušení dodávky elektrické proudu z jakýchkoliv důvodů, odpojení senzoru od převodníku a odpojení kabelu spojující počítač a převodník (dále jen: sériová linka). U testování těchto krizových stavů docházelo hned k několika komplikacím. Při odpojení napájecího kabelu, nebo výpadku elektrické energie docházelo k padání samotné aplikace a k její nefunkčnosti. Pokud by byl odpojen kabel se senzory, aplikace by rovněž spadla, jelikož daný senzor nebyl vůbec nalezen. Tím pádem nemohla být naměřená teplota uložena k sériovému číslu daného senzoru a tím ani nedocházelo ke kontrole správnosti uložení sériových čísel. Když byla odpojena sériová linka, došlo ke stejné chybě jako u předchozí krizové situace. Těmto krizovým stavům bylo nutné předejít a to tak, že se museli odstranit chyby, které vznikaly. Při padání aplikace totiž docházelo k markantní ztrátě naměřených dat. Tyto problémy se podařilo odstranit pomoc základního ošetřujícího příkazu jazyka C#. Po použití ošetřujících příkazů jazyka C# bylo spuštěno opětovné testování v krizových situacích. Ve všech případech krizových stavů se prokázalo, že aplikace po použití daných příkazů nepadá, ale jsou pouze vypsány chybové hlášky. A všechna dosud naměřená data se uloží do předem stanovených tabulek Excel.

V další fází testování bylo zapotřebí otestovat danou aplikaci v reálných podmínkách a podrobit jí tak zátěžovému testu. Toto testování bylo nejprve prováděno na kabelu o délce 60 metrů

a počtu 20 senzorů. Na této délce byla aplikace plně funkční a vše probíhalo bez problémů. Data byla uložena do tabulek Excel, které se nacházely na pevném disku ve složce dokumenty, pojmenovány jako „Měření_datum_čas“. Testování aplikace bylo prováděno také na 150m kabelu o počtu 15 senzorů. V této délce a při takovém počtu senzorů aplikace nefungovala správně. Docházelo k tomu, že nebyl nalezen žádný senzor. Při dalším opakovaném pokusu o nalezení byl nalezen jen určitý počet senzorů, který ale neodpovídal počtu senzorů na daném kabelu. Aplikace zaznamenávala data jen zlomkovitě a docházelo tak k neúplnému a nesprávnému ukládání naměřených hodnot teploty do tabulek Excel. Po tomto neúspěšném testování bylo prokázáno, že tato chyba nebyla chybou aplikace.



Obrázek 26. Tabulka Excel – výsledky testovacího měření



Obrázek 27. Okno Grafy – průběhy teplot testovacího měření

6. Závěr

Cílem této bakalářské práce bylo vytvořit vizualizační program pro sběr dat a ukládání naměřených hodnot do předem specifikovaných tabulek Excel. Zaměřil jsem se na podrobný popis struktury programu. Důležité bylo vizualizační program otestovat a následně vyzkoušet jeho funkčnost v reálných podmínkách na hlubinných podzemních vrtech.

Vytvořil jsem grafickou vizualizaci, která umožňuje změření teplot v hlubinných podzemních vrtech. Tento program tedy získává informace o teplotě pomocí digitálních čidel DS18B20. Dále tyto naměřené hodnoty zaznamenává do grafové komponenty, tabulky Excel a do tabulkové komponenty Visuál Studia. Pro účely přehlednějšího zaznamenávání naměřených teplot byla vytvořena dvě speciální okna. V prvním se nachází již zmíněná grafová komponenta, na které se vykreslují grafy naměřených hodnot v reálném čase. V druhém okně se tytéž hodnoty zaznamenávají do tabulek. Mimo tyto speciální okna jsem vytvořil další okno s nápovědou pro ovládání programu. Tato vizualizace byla vytvořena pro standardní osobní počítače a přenosné osobní počítače, ale důležitou součástí daného počítače musí být komunikační rozhraní sériového portu. Nebo můžeme použít počítač s komunikačním rozhraním USB a zasunout do něj převodník, čímž získáme plnohodnotný sériový port.

Výhodou tohoto vytvořeného vizualizačního programu je, že při získávání dat není zapotřebí PLC ani jiných zařízení pro řízení převodníku. Díky jednoduchosti vizualizačního programu se stačí připojit kabelem k převodníku. Spustit program, následně nastavit potřebné parametry a spustit měření. Po tomto procesu se dostaví výsledek v podobě souboru v tabulkách Excel uložených na disku osobního počítače. Hlavní výhodou tak zůstává flexibilita, přizpůsobivost a jednoduchost programu samotného. Ten totiž můžeme nainstalovat na jakýkoliv osobní počítač a výsledky následně zaznamenat na jakékoliv přenosné médium. Nevýhodou však zůstává časová omezenost vizualizačního programu. Nehodí se totiž k dlouhodobému měření a zaznamenávání hodnot teploty z hlubinných podzemních vrtů z důvodu jednoduchosti a praktičnosti tohoto vizualizačního programu. K těmto účelům je lepší použít propracovanější systémy určené ke sběru a archivaci dat.

Pro zjištění výsledků bylo nutné provést nezbytné testy a uvést vizualizační program do reálného provozu. Při prvním testování v domácích podmínkách se ukázalo, že je program funkční. Vykazoval jen drobné chyby, které jsem později odstranil. Ty nastaly neočekávaně a vyhodnotil jsem je, jako chyby v krizových situacích. Později jsem přistoupil k druhé fázi testování v laboratorních podmínkách. Zde se nacházelo podstatně větší množství senzorů na delším kabelu. Vyskytly se chyby na nejdelších kabelech, které jsem měl k dispozici.

Tento vizualizační program jsem vytvořil za účelem přehledného ukládání a jednoduššího získávání dat naměřených v hlubinných podzemních vrtech. Tento projekt měl usnadnit odečítání teplot z hlubinných podzemních vrtů. Vytvořením této podstatně jednoduší vizualizace jsem tak dosáhl větší dostupnosti programu.

7. Literatura

[1] HW server představuje – Sériová linka RS-232. [online]. revize 12. 12. 2005 [cit. 2012-12-15]. <http://www.hw.cz/rozhrani/hw-server-predstavuje-seriova-linka-rs-232.html>

[2] User Manual ADA-101W. [online]. 2001 [cit. 2012-11-10]. < http://cel-mar.pl/files/io/io_ada-101w_en.pdf>

[3] DS18B20 datasheet. [online]. 2001, revize 2008 [cit. 2012-12-17]. <http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf>

[4] LÁNÍK, Vladimír. MicroLan - A jde to i s jedním vodičem!. [online]. 2005 [cit. 2012-10-17]. <http://www.hw.cz/Rozhrani/ART1240-MicroLan---A-jde-to-i-s-jednim-vodicem.html>

[5] MALÝ, Martin. Sběrnice 1-Wire™. [online]. 2004 [cit. 2012-10-17]. <http://www.hw.cz/Rozhrani/ART1240-MicroLan---A-jde-to-i-s-jednim-vodicem.html>

[6] Co je knihovna DLL?. [online]. revize 2011 [cit. 2012-12-20]. <http://support.microsoft.com/kb/815065/cs>

[7] C sharp. [online]. revize 7. 3. 2013 [cit. 2013-4-15]. <http://cs.wikipedia.org/wiki/C_Sharp>

[8] Microsoft Visual Studio. [online]. revize 14. 4. 2013 [cit. 2013-4-15]. <http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio>

[9] PortBox2. [online]. revize 2012 [cit. 2013-1-1]. <http://www.hw-group.com/products/portbox/PortBox2_cz.html>

[10] Modems: Theory of Operation. [online]. [cit. 2013-1-1]. <http://docstore.mik.ua/orelly/other/puis3rd/0596003234_puis3-chp-10-sect-1.html>

[11] NAGEL, Christian, Bill EVJEN, Jay GLYNN, Karli WATSON, Morgan SKINNER a Allen JONES. *C# 2005: Programujem profesionálně*. 1.vydání. Brno: Computer Press, 2006. ISBN 80-251-1181-4.

8. Seznam příloh

PŘÍLOHA I - Bod zadání bakalářské práce č.7
PŘÍLOHA II - Ukázka kódu metody nalezení senzorů
PŘÍLOHA III - Ukázka kódu metody měření teploty
PŘÍLOHA IV - Elektronická příloha (CD)

PŘÍLOHA I

Abstrakt

Cílem této bakalářské práce je zabývat se návrhem a následnou realizací grafické vizualizace ve vývojovém software Visual Studio společnosti Microsoft, určenou pro aplikaci na měřící systém s digitálními čidly teplot DS18B20 od výrobce Dallas Semiconductor-Maxim, sloužící k zobrazování reálného stavu měření, korektnosti komunikačního spojení mezi připojenými čidly umístěnými v podzemních vrtech, připojených přes komunikační sběrnici 1-Wire na převodníky signálu a z něj přes komunikační rozhraní RS-232 k vyhodnocovacímu zařízení (počítač). Vizualizace je následně použita pro prezentaci naměřených dat a jejich převod do specifického formátu k archivaci.

Vizualizace je v současnosti velmi rozšířená například v průmyslu, a to především v automobilovém, stavebním, strojírenském, dále pak v medicíně, geografii apod. Setkáváme se s ní zejména ve velkých provozech, ale taky v menších. Díky ní můžeme v reálném čase sledovat a řídit technologické procesy, monitorovat stavy procesů a kontrolovat jeho kvalitu. Vizualizace nám dává také možnost efektivně reagovat na alarmové stavy a hlášení. Její nedílnou součástí je i možnost uložení dat. Vizualizace poskytuje několik výhod, jednou z hlavních je ovládání technologického procesu na dálku například pomocí internetu.

Princip této úlohy bude potřeba důkladně analyzovat z pohledu fyzické a softwarové realizace a následně vyřešit řadu klíčových bodů syntézy pro zajištění správnosti a funkčnosti výsledného řešení této problematiky. V případě fyzické analýzy by se měla řešit otázka použitých zařízení v měřícím řetězci, jejich komunikačních rozhraní včetně vhodného zapojení sběrnice a jejich vzájemná fyzická kompatibilita. Z hlediska analýzy správného postupu softwarového řešení a následné syntézy zde bude řešení komplikovanější a bude potřeba provézt řadu návrhu metodiky řešení pro jednotlivé specifické části celého měřicího systému a jeho vyhodnocovacího procesu. To zahrnuje řešení vzájemné komunikace zařízení na komunikačních kanálech použité sběrnice, zajištění navázání spojení a správnosti přenosu dat mezi zařízeními, řešení metodiky řízení jednotlivých zařízení, řešení metodiky sběru měřených dat a jejich analýzy a v neposlední řadě archivaci těchto dat. To vše je potřeba navrhnout tak, aby tento systém byl určitým způsobem závislý na uživateli z venčí.

Budeme-li uvažovat o topologii tohoto systému, měli bychom najít řešení softwarového spojení samotných teplotních čidel Dallas, jejich komunikaci s převodníky signálu a následném přenosu signálu do vyhodnocovacího zařízení.

Realizovaný systém by měl být schopen analyzovat celou strukturu měřícího řetězce pro zaručení bezchybného přenosu dat mezi jednotlivými elementy, případně na tyto chyby upozornit a následně získat veškerá potřebná informační data a měřená data z teplotních čidel a digitálních převodníků, která by měla být důležitá pro obsluhu systému a další zpracování.

Zde bude předpoklad vytvoření vhodného návrhu metodiky vyhodnocování správného chodu celého systému.

Realizovaný systém by měl mít určitou grafickou podobu, založenou na podmínce splnění několika zásad a to přehlednosti, funkčnosti a ergonomii práce s obsluhou. Představou je vytvoření grafického okna, které by bylo rozděleno do několika sektorů, přičemž každý z nich bude mít svůj specifický účel pro prezentaci určitých dat k uživateli.

Aplikace bude silně závislá na interakci s uživatelem, na jeho požadavcích, nastavení vstupních parametrů pro realizaci získávání dat z následného měření. Ovládání aplikace by však mělo splňovat zásadu intuitivního prostředí pro uživatele, ten by měl mít možnost konfigurovat postup nastavení a navázání komunikace s jednotlivými zařízeními v měřícím řetězci. Uživatel dostane také možnost parametrizovat si podmínky a dobu pro následné automatizované získávání měřených dat pro jejich archivaci, tzn., nastaví podmínky, po jakou dobu a s jakým vzorkováním dat by systematické měření mělo probíhat.

Aplikace bude schopna archivovat prezentovaná data do textových polí, které bude možno převést a uložit do tabulkového formátu jako je například Microsoft Excel. Dále se naměřené hodnoty zaznamenají v grafické podobě, jako graf naměřených dat. Předpoklad je, že by archivovaná data mohl uživatel sledovat přímo v aplikaci.

Případné nežádoucí stavy a poruchy v měřícím řetězci, vzniklé špatným nastavením, popřípadě problémy se spojením s některým ze zařízení, by měly být aplikací a její kontrolní části softwaru odhaleny a zobrazeny v upozornění pro uživatele.

Po vyřešení všech technických nedostatků, bude nutné vytvořit samotný program a důkladně jej otestovat. Nejprve bude nutné vyzkoušet, zda je aplikace schopná se připojit k danému přechodníku na zvolený sériový port. Následně se musí vyzkoušet i její pozdější odpojení a zapojení. Tato skutečnost slouží k otestování toho, zda po připojení k převodníku dojde k zobrazení informace o připojení na příslušný sériový port ve sloupci označeným jako Info. K testování bude nejprve použit měřící řetězec zapojen v následujícím pořadí: počítač, následoval převodník a poté samotné senzory v počtu 3 kusů na délce kabelu zhruba 5 metrů.

Poté program otestujeme v krizových situacích. Krizovými situacemi je myšleno odpojení napájecího kabelu nebo přerušení dodávky elektrické proudu z jakýchkoliv důvodů, odpojení senzoru od převodníku a odpojení kabelu spojující počítač a převodník (dále jen: sériová linka). U testování těchto krizových stavů by mělo docházet hned k několikati chybám, které by se měli narušovat chod samotného programu a zabraňovat ukládání naměřených dat do tabulek Microsoft Excel. Dalším cílem je tyto chyby odstranit a zajistit tak funkčnost aplikace i v těchto krizových situacích. Tyto problémy by se mělo dařit odstranit pomocí základního ošetřujícího příkazu jazyka C#. Po použití ošetřujících příkazů jazyka C# bude spuštěno opětovné testování v krizových situacích. Ve všech případech krizových stavů by se měly chyby ztrácet a pouze se vypsat ve sloupci Info.

Dále bude potřeba aplikaci podrobit zátěžovým testům přímo v reálném provozu. Testy budeme provádět v laboratořích VŠB-TU Ostrava.

Mezi výhody vytvořeného programu bude patřit především to, že při získávání dat nebude zapotřebí PLC ani jiných zařízení pro řízení převodníku. Předpoklad je, že díky jednoduchosti aplikace se bude možné připojit jen pomocí kabelu k převodníku, následně spustit program s předem nastavenými potřebnými parametry a měřit. Po skončení celého procesu se naměřená data budou zapisovat do tabulek Excel uložených na disku osobního počítače. Hlavní výhodou tak zůstává flexibilita, přizpůsobivost a jednoduchost programu samotného. Ten je totiž možné nainstalovat na kterýkoliv osobní počítač a naměřené hodnoty archivovat na přenosná média. Nevýhodu bude zřejmě představovat časová omezenost vizualizačního programu. Nehodí se totiž k dlouhodobému měření a zaznamenávání hodnot teploty z hlubinných podzemních vrtů z důvodu jednoduchosti a praktičnosti tohoto vizualizačního programu.

Abstract

The aim of this thesis is to discuss the design and subsequent implementation of a graphical visualization in Visual Studio development software company Microsoft, designed for application to the measurement system with digital temperature sensors DS18B20 manufacturer from Dallas-Maxim Semiconductor, used to display real-state measurements, correctness of communication between connected sensors placed in underground boreholes, connected by a communication bus 1-Wire the signal converters and out through the RS-232 to a suitable device (computer). Visualization is subsequently used for presenting measurement data and converts them into a specific format for archiving.

Visualization is now widespread example in industry, primarily in the automotive, construction, engineering, as well as in medicine, geography, etc. We meet with her, especially in large plants, but also in the smaller ones. With it we can real-time monitoring and control of technological processes, monitor the process states and control its quality. Visualization also gives us the ability to effectively respond to alarm conditions and reporting. Its integral part is the ability to store data. Visualization provides several advantages, one of the key is to control the process at a distance such as the Internet.

The principle of this role will need to be carefully analyzed in terms of physical and software implementation and subsequently address a number of key points of synthesis to ensure the accuracy and functionality of the final solution of this problem. In the case of physical analysis should address the question of the devices used in the measurement chain, their communication interfaces including appropriate involvement of the bus and their mutual physical compatibility. According to an analysis of good practice software solution and subsequent synthesis solution here is complicated and will need to perform a number of design methodology solutions for each specific part of the measuring system and the evaluation process. This includes solutions of communication equipment communication channels used bus, providing connection and accuracy of data transfer between devices, solutions management methodology of device solutions measured data collection methodology and analysis and finally archiving the data. All this needs to be designed so that the system was in some way dependent on the user from the outside.

If we think about the topology of the system, we should find a solution to the software connection alone temperature sensors Dallas, their communication with the signal converters and the subsequent signal to the sensing device.

Realized system should be able to analyze the whole structure of the measuring chain to guarantee error-free data transmission between elements or to highlight these errors and then get all the necessary information and data measured data from temperature sensors and digital converters, which should be important for the system for further processing. You will be prerequisite for the creation of a suitable design methodology evaluation proper operation of the system.

The realized system should have a graphic form, based on the condition that several principles and clarity, functionality and ergonomics of the operator. The idea is to create a graphical window, which would be divided into several sectors, each of which will have its specific purpose for the presentation of specific data to the user.

Applications will be strongly dependent on the interaction with the user on the requirements set of input parameters for implementing the acquisition of data from subsequent measurements. Control applications, it should comply with the principles of intuitive environment for users who should be able to configure how to set up and establish communication with the individual devices in the measuring chain. The user gets the possibility to parameterize the conditions and the time for subsequent automated acquisition of measurement data archiving, ie. sets conditions for how long, and what sampling data by systematic measurements should be performed.

Applications will be able to archive the data presented in text fields that can be converted and saved in spreadsheet format, such as Microsoft Excel. Furthermore, the measured values recorded in graphical form, as a graph of measured data. The assumption is that the archived data the user can watch directly in the application.

Possible adverse conditions and disorders in the measuring chain, caused by incorrect settings, or connection problems with any of the equipment should be application and its control of the software detected and displayed in a notice to the user.

After you resolve any technical deficiencies will need to create the actual program and test it thoroughly. First, you will need to test whether the application is able to connect to that transgressive to the selected serial port. Subsequently, the test and its subsequent disconnection and connection. This fact is used to test whether the connection to the converter will display the connection information to the appropriate serial port in the column labeled as Info. The testing will be initially used measuring chain involved in the following order: the computer, and then followed the converter itself in a number of sensors 3 pieces at a length about 5 meters.

Then test the program in crisis situations. Crisis situations are meant unplugging the power cord or interruption of electrical power for any reason, disconnect the sensor from the transmitter and disconnect the cable connecting the PC and converter (hereinafter: the serial line). For testing the state of emergency should occur immediately to the umpteenth errors that should interfere with the operation of the program itself and prevent storage of measured data into spreadsheets Microsoft Excel. Another goal is to rectify these errors and ensure application functionality in these crisis situation. These issues should be removed with a flourish attending basic command of language C #. After attending commands using the C # language will be re-triggered testing in emergency situations. In all cases, crisis situations, errors should lose and only to dump in the Info column.

Furthermore, the application will need to undergo stress tests in the real operation. Tests will be carried out in laboratories VŠB-TU Ostrava.

The benefits created by the program will be mainly that the retrieval will be required PLC or other control device converter. The assumption is that due to the simplicity applications we can only connect via cable to the converter, then run a program with predefined parameters and necessary measure. At the end of the process the measured data will be written into Excel spreadsheets stored on a personal computer. The main advantage remains flexibility, adaptability and simplicity of the program itself. It is not possible to install on any personal computer and the measured value archive on removable media. The disadvantage is likely to be a time limitation of visualization software. Not applicable because the long-term measurement and recording of temperature from deep underground wells because of the simplicity and practicality of the visualization program.

PŘÍLOHA II

Ukázka kódu metody nalezení senzorů

```
/// <summary>
   /// Metoda pro nalezení senzorů
   /// </summary>
   /// <returns></returns>
   public List<Sensor> FindAllDevices()
   {
      int count = 0;
      List<Sensor> sensors = new List<Sensor>();//vytvoření nového listu senzorů
      Sensors.Sensor sens = null;
      // zakladní nastavení adaptéru
      vPortAdapter.BeginExclusive(true);
      vPortAdapter.SetSearchAllDevices();
      vPortAdapter.TargetAllFamilies();
      vPortAdapter.Speed = OWSpeed.SPEED_REGULAR;

      if (vPortAdapter != null)//otestování připojení adaptéru
      {
         byte[] address = new byte[8];
         //otestuje přítomnost prvního zařízení
         if (vPortAdapter.GetFirstDevice(address, 0))
         {  //cyklus pro přidavání seriových čísel senzorů do listu
            do
            {
               sens = new Sensor();//vytvoří nový senzor
               sens.SerialNumber = (byte[])address.Clone();//klonování adresy
               sensors.Add((Sensor)sens.Clone());//přidá senzor do senzorů
               address = new byte[8];

               if (NewInfo2 != null)
               {
                 NewInfo2(sens.GetSerialNumber().ToString("X2"));//vypíše do infa sérivé
                 číslo v hexa formátu
               }
            }
            while (vPortAdapter.GetNextDevice(address, 0));
         }
         vPortAdapter.EndExclusive();//ukončení použití adaptéru

      }
      else
      {
      }
      return sensors;

   }
```

PŘÍLOHA III

Ukázka kódu metody měření teploty

```
/// <summary>
/// Metoda měření teploty
/// </summary>
/// <param name="sens"></param>
public int ReadTemperature(List<Sensor> sens)
{
   int pocet,retval=0;
   try
   {
      if (NewInfo != null)//otestuje zda se událost provedla
      {
         NewInfo("Vola se cteni teploty");
      }

      if (vPortAdapter != null)//testování připojení adaptéru
      {
         pocet = 0;

         // zakladní nastavení adaptéru
         vPortAdapter.BeginExclusive(true);
         vPortAdapter.Speed = OWSpeed.SPEED_REGULAR;
         vPortAdapter.TargetAllFamilies();
         //vPortAdapter.Reset();
         vPortAdapter.SetSearchAllDevices();

         vPortAdapter.Reset();
         vPortAdapter.PutByte(0xCC);
         vPortAdapter.PutByte(0x44);//příkaz pro měření teploty
         System.Threading.Thread.Sleep(1200);//pauza pro převod teploty

         byte[] address = new byte[8];
         //otestuje přítomnost prvního zařízení
         if (vPortAdapter.GetNextDevice(address, 0))
         {  //cyklus pro přidavání naměřených hodnot teploty
            do
            {
               pocet++;

               byte[] packet = new byte[9];
               packet[0] = (byte)(0x55);
               Array.Copy(address, 0, packet, 1, 8);
               vPortAdapter.DataBlock(packet, 0, 9);//pošle se příkaz MATCH ROM

               // reset 1-Wire
               OWResetResult rst2 = vPortAdapter.Reset();
```

```
  Array.Copy(address, 0, packet, 1, 8);
  vPortAdapter.DataBlock(packet, 0, 9);


  byte[] packetD = new byte[10];
  for (int i = 0; i < 10; i++)
  {
     packetD[i] = 0xFF;
  }
  packetD[0] = (byte)(0xBE);
 vPortAdapter.DataBlock(packetD, 0, 10);//pošle se příkaz READ
SCRATCHPAD

  if ((CRC8.compute(packetD, 0, 7)) - packetD[7] == 0 || true)//otestuje
  správnost přijatých dat pomocí crc
  {
     int invLSB, invMSB;
     int teplota;
     double vystep = double.MinValue;
     switch (packetD[5])//volba rozsahů
     {
        case 0x7F://12bitů
           if (packetD[2] < 8)//otestuje zda se jedna o kladnou hodnotu
           {
              // posun o 8 bitů doleva
              teplota = ((int)(packetD[2])) << 8;
              teplota = teplota + ((int)(packetD[1]));//součet horní a dolní teploty
              // vypočet teploty
              vystep = (double)(teplota * 0.0625);
           }
           else
           {
              // převrácení bytu packetD.
              invLSB = packetD[1] ^ 0xFF;
              invMSB = packetD[2] ^ 0xFF;
              // posun o 8 bitů doleva převraceného bytu
              teplota = invMSB << 8;
              teplota = teplota + invLSB;
              // vypočet teploty
              vystep = (double)(teplota * (0.0625 * (-1)));
           }
           break;

        case 0x5F://11bitů
           if (packetD[2] < 8)
           {
              // posun o 8 bitů doleva
              teplota = ((int)(packetD[2])) << 8;
              teplota = teplota + ((int)(packetD[1]));//součet horní a dolní teploty
              teplota = teplota >> 1;
```

```
            // vypočet teploty
            vystep = (double)(teplota * 0.5);
        }
        else
        {
            // převrácení bytu packetD.
            invLSB = packetD[1] ^ 0xFF;
            invMSB = packetD[2] ^ 0xFF;
            // posun o 8 bitů doleva převraceného bytu
            teplota = invMSB << 8;
            teplota = teplota + invLSB;
            teplota = teplota >> 1;
            // vypočet teploty
            vystep = (double)(teplota * (0.5 * (-1)));
        }
        break;

    case 0x3F://10bitů
        if (packetD[2] < 8)
        {
            // posun o 8 bitů doleva
            teplota = ((int)(packetD[2])) << 8;
            teplota = teplota + ((int)(packetD[1]));//součet horní a dolní teploty
            teplota = teplota >> 2;
            // vypočet teploty
            vystep = (double)(teplota * 0.25);
        }
        else
        {
            // převrácení bytu packetD.
            invLSB = packetD[1] ^ 0xFF;
            invMSB = packetD[2] ^ 0xFF;
            // Identifikace bitů bytu inv.
            teplota = invMSB << 8;
            teplota = teplota + invLSB;
            teplota = teplota >> 2;
            // vypočet teploty
            vystep = (double)(teplota * (0.25 * (-1)));
        }
        break;

    case 0x1F://9bitů
        if (packetD[2] < 8)
        {
            // posun o 8 bitů doleva
            teplota = ((int)(packetD[2])) << 8;
            teplota = teplota + ((int)(packetD[1]));//součet horní a dolní teploty
            teplota = teplota >> 3;
            // vypočet teploty
            vystep = (double)(teplota * 0.125);
```

```
                    }
                    else
                    {
                        // převrácení bytu packetD.
                        invLSB = packetD[1] ^ 0xFF;
                        invMSB = packetD[2] ^ 0xFF;
                        // posun o 8 bitů doleva převraceného bytu
                        teplota = invMSB << 8;
                        teplota = teplota + invLSB;
                        teplota = teplota >> 3;
                        // vypočet teploty
                        vystep = (double)(teplota * (0.125 * (-1)));
                    }
                    break;
                }


                // vyčtená hodnota se zapíše do senzoru s příslušnou adresou
                foreach (Sensor item in sens)
                {

                    if (item.CompareTo(address) == 0)
                        item.Records.Add(new Record(DateTime.Now, vystep));
                }
            }
            else
            {   //v případě chyby crc se vypíše hlaška
                if (NewInfo != null)
                {
                    NewInfo("Špatný CRC");
                }

            }

        }
        while (vPortAdapter.GetNextDevice(address, 0));
    }

  }
  retval = 0;//v případě připojení hodnot je  0
 }
 catch
 {
  retval = 1;//v případě připojení hodnot je  1
 }
 return retval;// vrací hodnotu

}
```

PŘÍLOHA IV

Elektronická příloha (CD)

- Visual studio - Program „ Měření teploty 1-Wire“
- DS18B20 Datasheet.pdf
- DS2480B Datasheet.pdf
- ADA-101W User Manual.pdf
- Bakalářská práce Lukáš Šilbach.pdf